TEX 入門

桂田 祐史 (かつらだ まさし)

2012 年 8 月 16 日, 2015 年 2 月 17 日

最新版は http://nalab.mind.meiji.ac.jp/~mk/labo/text/tex2014.pdf

目 次

TeX をどう使いこなすかは、利用しているコンピューター環境にもよるし、時の経過につれて少しずつ変化しているので、年度毎に作り直すことにしました。最初に作った版は数学科学生の情報処理教室での利用が念頭にありましたが、2014 年版からは現象数理学科の MacBook での利用を前提にすることにします。

1 TeX とは？

(駆け足で説明する。)

TeX は組版ソフトである

TeX は、著名なコンピューター科学者であるドナルド・クヌース (Donald Knuth[1], ウィキペディア[2] にも載っています, "The Art of Computer Programming" シリーズが有名です) が開発した**文書整形システム** (組版[3] システム) です (最初のバージョンは 30 年以上前に作られました)。TeX は日本では、「てっく」または「てふ」と呼ばれることが多いです[4]。

当初、数式を含む英語の文章を清書することを目的に、従来の組版技術の歴史を入念に調べた上で、それをコンピューター上で実現することを目標に開発されたそうです。

[1] http://www-cs-faculty.stanford.edu/~knuth/

[2] http://ja.wikipedia.org/wiki/ドナルド・クヌース

[3] 組版 (くみはん) とは、文字や図版などの要素を配置して、紙面を構成することで、もともとは活版印刷において、活字を組み上げることから来ている。

[4] 「てっくす」とは読みません。ちなみに TeX の解説書に "Joy of TeX" という本があって、それは英語圏の国では有名な本のパロディだったそうです。昔、テレビで深夜映画を見ていたら、元ネタの本が出て来て、思わず見入ってしまいました。

ワープロ (ワードプロセッサー・ソフトウェア) と比べると[5] 一長一短ありますが、特に長い論文や書籍のような文章を組版するには向いているとされています。

TEX はフリーソフトである

Knuth 自身は TEX に関する情報を完全に公開していて (書籍になっています)、ソフトウェアは無償で利用することができます。また、多くのボランティアの活動により、TEX を補助、発展させるためのソフトウェア、データもほとんどは無償で利用可能です。例えば、TEX 本体や周辺ソフトウェアの C 言語への変換、画面表示用ドライバー (プリヴューアーと呼ばれます)、印刷用ドライバー、PDF への変換ソフトウェア、日本語対応、ラテン文字&数式記号のフォント、日本語フォント (やそれを利用する仕組み)、Windows 環境への移植、インストーラーなどなど。これら成果物は大抵はインターネットから無償で入手できます。

上はソフトウェアについて書きましたが、そういうソフトウェアを使いこなすための情報もインターネット上で入手できます。一般にネットで入手できる情報は玉石混交の場合が多いですが、TEX に関する情報は良いものが多いと感じています。

TEX は数学の世界では標準である

数学者村では、標準の文書作成ソフトウェアです。理工系の多くの分野で利用されていますが、それだけでなく文系の研究者が利用した例もあります (発音記号や、ややマイナーな言語などを扱う場合)。

TEX で高品位の文書が作成できる

組版技術をしっかり研究した上で作られたものであるため、高品質な仕上がりが得られます。異なる環境下での再現性も抜群です (誰が何処で何を使って印刷しても同じ仕上がり — 同じフォントが使えれば、ですが。古いパソコンで出来たことが新しいパソコンでは出来るとは限らないし、その逆も当たり前、とは考えないこと。)。英語圏ではもちろん、日本でも理工系の多くの書籍 (中学高校の教科書や問題集なども含む) で採用されています。

TEX で作った文書は PDF にして配布が楽々

TEX 自身は文書の配布フォーマットとして適当ではありませんが (表示、印刷に専用のソフトウェアが必要なためです)、TEX で書いた文書は簡単に **PDF** (portable document format) に変換できるので、そうしてから配布すれば、相手が読めるだろうか、印刷できるだろうか、心配する必要はほとんどありません。

この授業では、TEX の一種である **LATEX** (正確にはその日本語対応版 pLATEX) を使ってもらうことにします。

[5] ワープロは WYSIWYG (What you see is what you get), つまり「画面に見えているものがそのまま印刷される」、「印刷される見栄えのまま画面で作業できる」で、TEX のようなコマンド形式のソフトウェアとは大きな違いがあります。

2　LaTeX 最初の一歩（TeXShop 版）

TeXShop を起動してみよう。色々な設定の仕方がありうるけれど、必ず出来そうなのは、Finder でアプリケーションから TeXShop をダブルクリックして起動する、というやり方である。

“名称未設定-1” という名前のついたウィンドウが出て来るはずで、キーボードから入力して、図 1 のようにしよう。

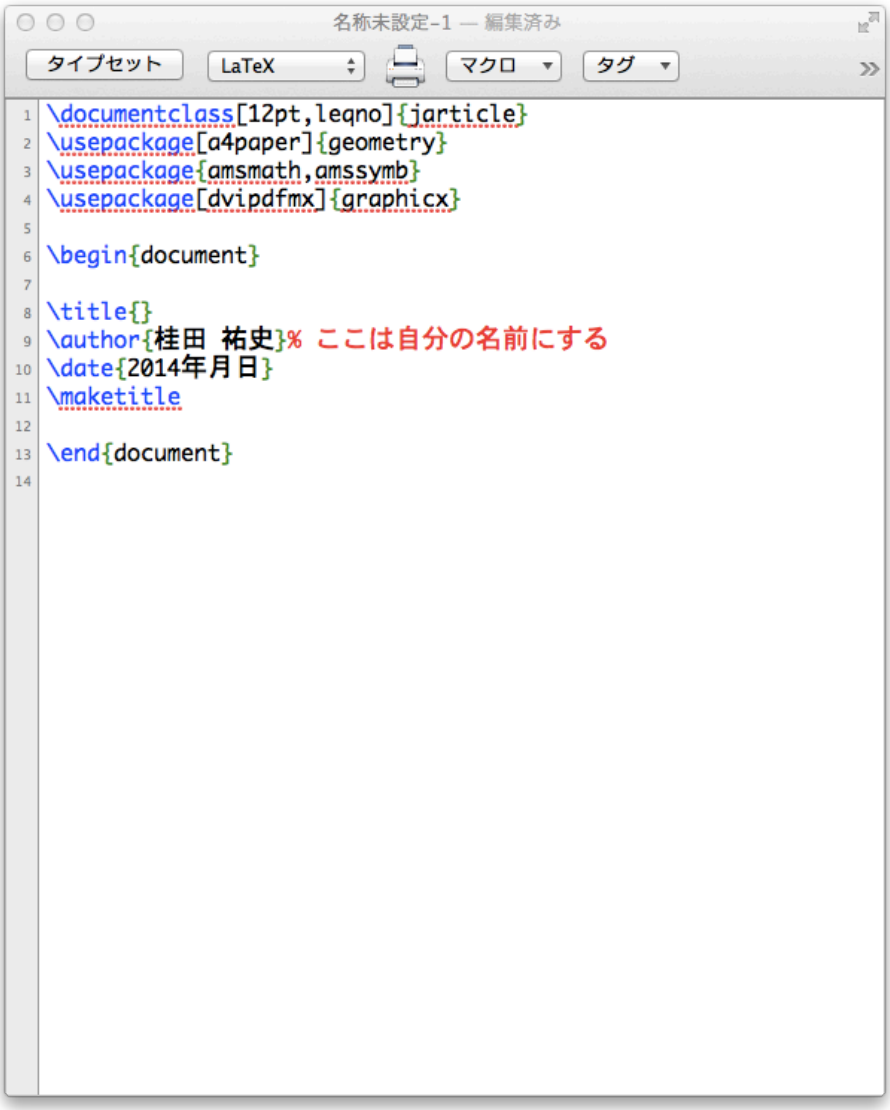


図 1: TeXShop にひな形を入力

[ファイル] メニューから [書き出す] を選択すると、書き出し名を尋ねられるので、適当な場所、適当な名前を指定する。自分で TeX 文書用のディレクトリィやゼミ授業のディレクトリィを準備して、そこに保存するのが良いが、ここでは書類ディレクトリィに保存する。

少し書き足してみよう。

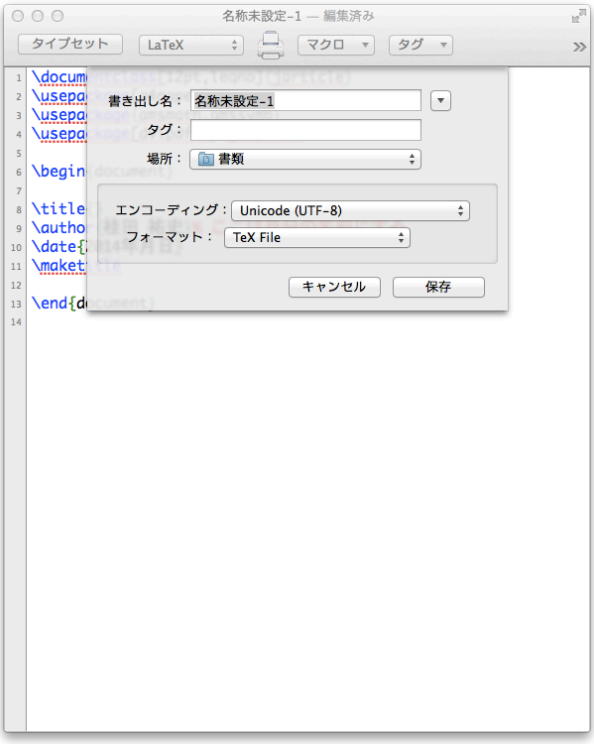


図 2: 書き出す時のウィンドウ — ▽ ボタン

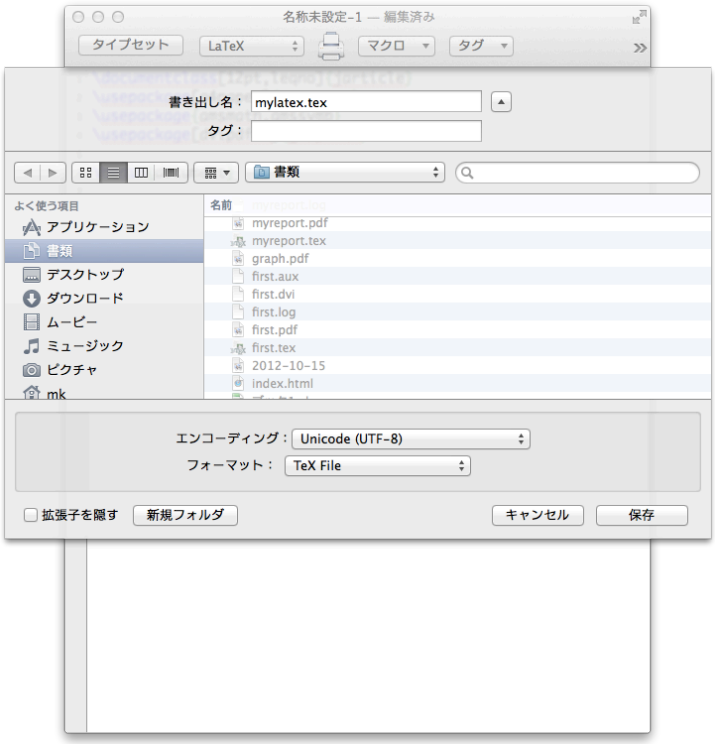


図 3: 書類フォルダに mylatex.tex という名前で書き出す

first.tex

```
\documentclass[12pt,leqno]{jarticle}
\usepackage[a4paper]{geometry}
\usepackage{amsmath,amssymb}
\usepackage[dvipdfmx]{graphicx}

\begin{document}
\title{初めての\TeX}
\author{桂田　祐史}% ここは自分の名前にする
\date{2014 年 6 月 6 日}
\maketitle

こんにちは。

\[
  \int_{-\infty}^{\infty} e^{-x^2} dx=\sqrt{\pi}.
\]
\end{document}
```

[ファイル] メニューの項目 [保存] を選び、myfirst.tex という名前で保存しよう (名前には “myfirst” とだけ入力すれば良い)。

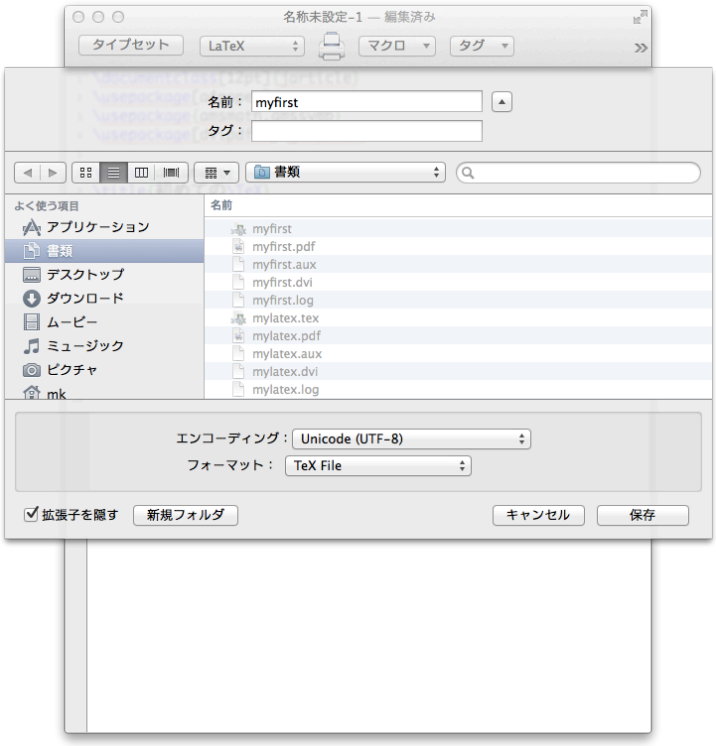


図 4: 保存する場所と名前を指定する

タイプセット ボタンを押すと、入力間違いがなければ、図 5 のようになる。

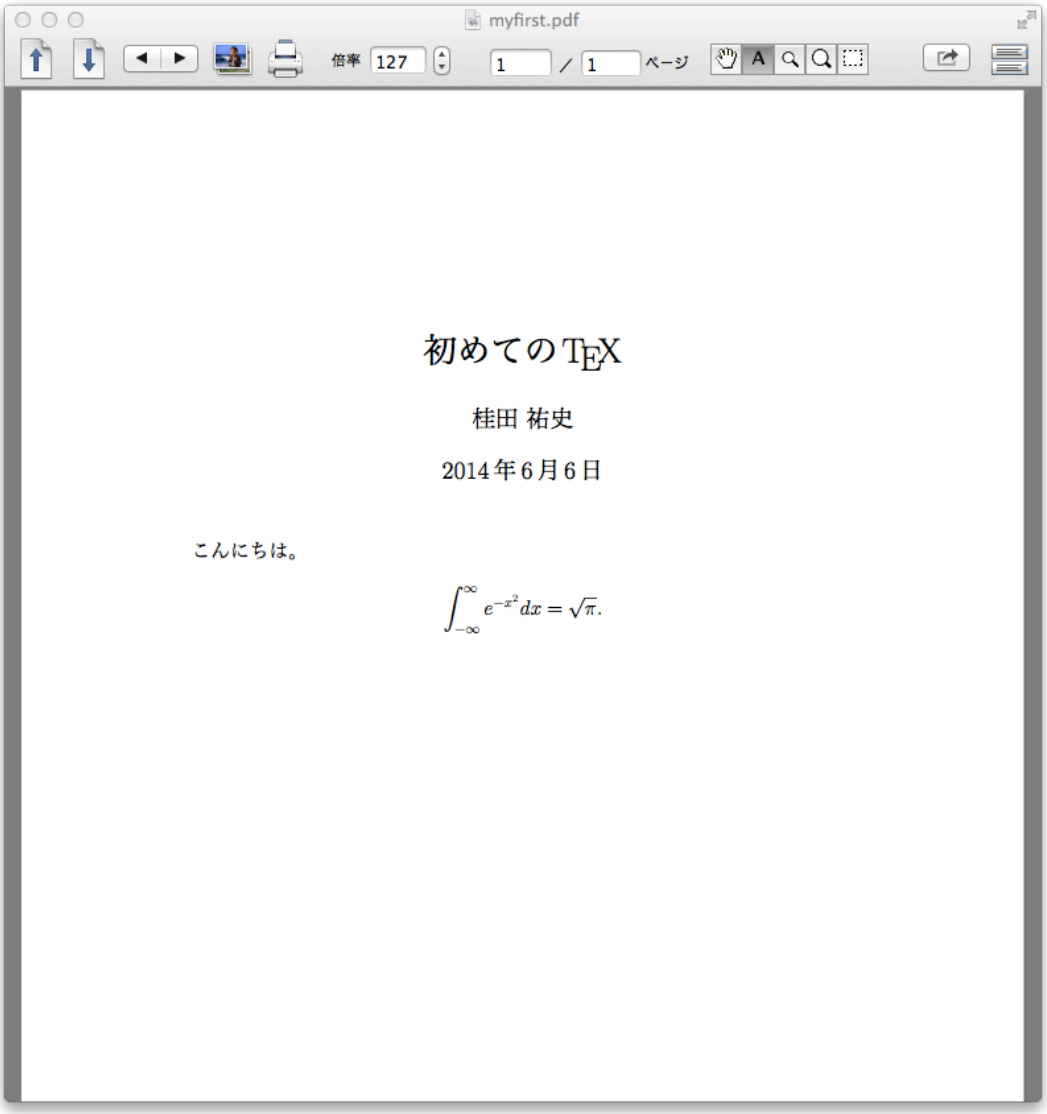


図 5: こんなふうなのが出て来れば成功

mylatex.tex をひな形として使いたければ、ターミナルから以下のように保存する。

```
cp ~/Documents/mylatex.tex ~/Library/TeXShop/Templates/
```

こうしておくと、次回 TeXShop を起動した時から、テンプレートから mylatex を選択すると、mylatex.tex の内容が挿入される。

3 LaTeX 最初のおさらい

3.1 LaTeX 文書の書き方

数式の書き方は置いておくとして、見本で使うような基本的事項を説明しておく。

LaTeX 文書で最低限必要なのは次の内容である。

```
\documentclass[12pt,leqno]{jarticle}% スタイルの指定
\begin{document}
\end{document}
```

レポート等では、タイトル、著者名、日付が必須なので、次のようなものが必要と思って良い。

```
\documentclass[12pt,leqno]{jarticle}% スタイルの指定
\begin{document}
% この行は注釈。次の 4 行でタイトル、著者名、日付を表示する
\title{レポート課題 X}
\author{1 年 2 組 99 番 桂田 祐史}
\date{2014 年 6 月 6 日}
\maketitle
\end{document}
```

特殊文字以外は \begin{document} と \end{document} の間に書けば表示される。

```
\documentclass[12pt,leqno]{jarticle}% スタイルの指定
\begin{document}
% この行は注釈。次の 4 行でタイトル、著者名、日付を表示する
\title{レポート課題 X}
\author{1 年 2 組 99 番 桂田 祐史}
\date{2016 年 6 月 6 日}
\maketitle

  ここにフツーの文字で書いたものは出力される。
\end{document}
```

実際には色々な記号が TeX の命令と解釈される特殊な文字となっている。プログラムなどを表示するには、verbatim 環境で利用するのが簡単である。

```
\documentclass[12pt,leqno]{jarticle}% スタイルの指定
\begin{document}
% この行は注釈。次の 4 行でタイトル、著者名、日付を表示する
\title{レポート課題 X}
\author{1 年 2 組 99 番 桂田 祐史}
\date{2014 年 6 月 6 日}
\maketitle

  ここにフツーの文字で書いたものは出力される。

% verbatim 環境の中は特殊文字であってもそのまま出力される。
\begin{verbatim}
#include <stdio.h>
int main(void)
{
  printf("Hello\n");
  return 0;
}
\end{verbatim}
\end{document}
```

3.2 TEX のための準備作業

現象数理学科学生向けの MacBook Air では、TeXShop で使うための準備作業は済んでいるはず (？)。どうもそうとは言い切れないようだ…

以下は自分でやろうという人のための情報である。

- Mac で TEX を使うには、MacTeX[6] と TeXShop[7] をインストールして、TeXShop をほんの少し設定するだけで OK (自分でやるなら「2014 年のコンピューターノウハウ (Mac)」[8] を参考にして下さい).
- 大抵のことは TeXShop から使えるけれど、ターミナルの中からコマンドを入力して使いたい場合は、適当に PATH[9] の設定をする必要がある。

  bash を使う場合は .profile の尻尾に

  ```
  export PATH=$PATH:/usr/local/texlive/2013/bin/x86_64-darwin
  ```

  tcsh を使う場合は .tcshrc の尻尾に

  ```
  set path=($path /usr/local/texlive/2013/bin/x86_64-darwin)
  ```

[6] http://www.tug.org/mactex/
[7] http://darkwing.uoregon.edu/~koch/texshop/texshop.html
[8] http://nalab.mind.meiji.ac.jp/~mk/knowhow-2014/knowhow-2014.html
[9] MacTeX 2013 の場合は、/usr/local/texlive/2013/bin/x86_64-darwin/

3.3 基本的な LaTeX の使い方

実は TeX は、大小様々なソフトの連携プレーであると言える。TeXShop は色々なソフトを呼び出すことで役目を果たしている。以下、TeXShop を使わない方法 (TeXLive に含まれるソフトを利用する) を説明する。

1. mylatex.tex を (テキストエデット, Emacs テキスト・エディターで) 開いて、適当な名前に変えて保存してから、編集 (執筆？) を始める。
   (試すなら、課題 X のために、kadaiX.tex というファイルを作ることを勧める。)

2. なんとか.tex を dvi ファイルに変換 (コンパイル？) するには、ターミナルで、

   ```
   bash-3.2$ platex なんとか.tex Enter
   ```

   とする (下線部を入力する, 以下繰り返さない)。

3. なんとか.dvi を表示 (プリビュー) するには、コマンド・プロンプトで

   ```
   bash-3.2$ pxdvi なんとか.dvi & Enter
   ```

   とする。

4. 人に渡す場合は、PDF ファイルにするとよい。なんとか.dvi を PDF ファイルに変換するには、コマンド・プロンプトで

   ```
   bash-3.2$ dvipdfmx なんとか.dvi Enter
   ```

   とする。これで なんとか.pdf が出来上がるはず。

4 LaTeX 文書 .tex の書き方 — 入門

4.1 最初に覚えるべきこと

- ファイル名の拡張子は ".tex" とする。
- TeX のコマンドには、先頭に「バックスラッシュ」 \ をつけるが、日本語環境では、「円記号」 ¥ として表示される場合が多い。「円記号」 ¥ も「バックスラッシュ」 \ も、(見栄えは違うけれど、文字コードは同じ (16 進数で 0x5c, 10 進数で 92) なので [10]) TeX のコマンドにとっては同じである。
- パーセント記号 % から行末までは注釈になる。
- いつでも書くことになりそうな次の内容は、mylatex.tex に書いておいた (自分の氏名などを書き足すと良いかもしれない)。

[10] 最近、この「常識」が通用しなくなるケースも出て来ました。今の Mac では、購入時の状態で、¥ キーを押して入力されるのは、「バックスラッシュ」 \ とは異なる、Unicode で用意された「円記号」 ¥ です。日本語入力システム (たとえば「ことえり」) の設定を変更しないと「バックスラッシュ」 \ が入力できません。

mylatex.tex

```
\documentclass[12pt,leqno]{jarticle}
\usepackage[a4paper]{geometry}
\usepackage{amsmath,amssymb}
\usepackage[dvipdfmx]{graphicx}

\begin{document}

\title{}
\author{桂田 祐史}% ここは自分の名前にする
\date{2014 年月日}
\maketitle

\end{document}
```

これを読み込んで、別名で保存する (あるいはテンプレートに登録しておいて呼び出す)、というやり方を勧める。

- タイトルをつけるには、

```
\title{はじめての \TeX}% タイトル
\author{桂田 祐史}      % 著者名
\date{2014 年 6 月 6 日 }    % 日付（省略すると組版した日になる）
\maketitle              % これでタイトルを表示する
```

(date を省略すると、組版 (タイプセット) 実行時点の年月日が元号で表示されるが、西暦にするには、\ 西暦 コマンドを用いる。)

```
\西暦
\title{はじめての \TeX}
\author{桂田 祐史}
\maketitle
```

(レポートなどを書く際は、最初に締切日を \data{} に書き込んでおいて、それを % で注釈にしておき、提出するときに注釈を外す、というやり方をすすめたい。)

- \begin{document} から \end{document} までの間に、ローマ字、数字などの “フツーの字” で書くとそのまま文書に入力される。いわゆる記号は注意が必要である。

細かい話: 記号について

まず、そのまま入力&表示できるものとして、

```
!   "   ’   (   )   -   =       @   [   ]   +   *       :   ?   ,   .
```

がある (マイナス - は、1 文字の場合、2 文字連続の場合、3 文字連続の場合で、それぞれハイフン -, en-dash –, em-dash — となるので、そのまま入力できるものと考えるべきではないかもしれない。もっとも通常、マイナスは数式中に現われるものだから「$-$ と書く」と覚えるべきかも。)。

一方、シャープ # などは、そのまま入力したのではダメで、これはバックスラッシュ \ を前につけて \# と入力する必要がある (\ でエスケープする、という)。同様にエスケープする必要がある文字としては、

```
#   $   %   &   _   {   }
```

がある。

$ と $ で囲んで、数式モードで扱うべき文字としては

```
|   <   >   −
```

がある。

難しいのは次の 3 文字で、これを表示するには、(右側に書いた) 専用のコマンドを用いなければいけない。

```
~   \textasciitilde
\   \textbackslash
^   \textasciicircum
```

日本人専用の応急処置として、難しい文字の入力には漢字を使う、という手がある (やや幅広になってしまうけれど)。# $ % & _ { } | < > ～ ＼という感じで簡単。

4.2 改行と空白 (最低限の注意)

意外と難しいので[11]、ある程度 TEX の説明が進んでから詳しく説明する (と言って、例年さぼっています)。ここではごく基本的なことと、「予告」に止める。

- .tex の中にいくら空白を続けても、一つ空白を入れたのと同じで、小さな空きができるだけ。

  ```
  This is        a                                                    pen.
  ```

  は

  This is a pen.

  となる。空白を明示的に入力するために、\quad などのコマンドがある。数式モードでは微調整用のコマンドがたくさんある。

- 連続した改行は「空行」と呼び、パラグラフ (段落?) を変更するという意味になり (\par と同じ)、改行されて、次の文の先頭に空きができる (いわゆる段落先頭の字下げ (indentation))。連続した空行は 1 つの空行と同じことになる。

- 英文中の一つの改行 (空行でないもの) は、一つの空白と同等。日本語文中の一つの改行 (空行でないもの) は、無視される。(不正確な言い方だが…)

[11] 少し大げさなようだが、TEX の設計思想に係わることなので。

```
I
love
you.
弁慶がな
ぎなたを
```

は

I love you. 弁慶がなぎなたを

となる。

- 強制的な改行は \\ だが、初心者が使いたくなるケースの 95% は誤用である (卒研で君達の先輩を相手にしたときの経験則)。

「(数式・表でないところで) 強制改行は極力使わない」

と考えることを勧める。

4.3 文字の大きさと書体

実は結構複雑である。ここでは (9 割の要求に応えれば良いことにして) 簡単に済ませる。

4.3.1 文字の大きさ

文字の大きさを変えるには、以下のようなコマンドがある。

- \tiny
- \scriptsize
- \footnotesize
- \small
- \normalsize
- \large
- \Large
- \LARGE
- \huge
- \Huge

```
{\tiny a}
{\scriptsize a}
{\footnotesize a}
{\small a}
{\normalsize a}
{\large a}
{\Large a}
{\LARGE a}
{\huge a}
{\Huge a}
```

a a a a a a a a a a

もっと大きくしたい？

```
\usepackage[dvipdfmx]{graphicx}% graphicx が必要
\begin{document}
...

\scalebox{10.0}{a}
```

4.3.2 文字の書体の指定

最初のうちは、こういうことに凝らないことを勧めたいけれど。欧文書体の場合は

- \textrm{}　(普通の) ローマン体　abcABC
- \textit{}　イタリック体　*abcABC*
- \textsf{}　サンセリフ体　abcABC
- \texttt{}　タイプライター体　abcABC
- \textbf{}　ボールド体　**abcABC**
- \textsc{}　スモールキャピタル体　ABCABC
- \textsl{}　スラント体　*abcABC*

日本語の場合は、\textgt{} でゴシック、\textmc{} で明朝。普通は明朝なので、

太くしたいときは (欧文書体と同様に) \textbf{} を用いる

でも良いかもしれない。

```
桂田です。 \textgt{桂田です。} \textmc{桂田です。} \textbf{桂田です。}
```

桂田です。 **桂田です。** 桂田です。 **桂田です。**

(この例は、WWW では左から二番目がゴシック体で表示されない。)

5　簡単な数式

5.1　数式モード

数式は「数式モード」の中で書く。数式モードには次の二つがある。

1. 文中の数式 (**インライン数式**) は、ドル記号 `$` ではさんでかく。

```
ピタゴラスの定理から $a^2+b^2=c^2$ が成り立つ。
```

ピタゴラスの定理から $a^2+b^2=c^2$ が成り立つ。

2. 数式のみの行 (**ディスプレイ数式**) を作るには、色々な命令があるが、もっとも基本的なものは、`\[` と `\]` ではさむもので、例えば

```
ピタゴラスの定理から
\[
  a^2+b^2=c^2
\]
がなりたつ。
```

のようにすると

ピタゴラスの定理から
$$a^2+b^2=c^2$$
がなりたつ。

となる。式番号をつけるには equation 環境というものを用いて、

```
ピタゴラスの定理から
\begin{equation}
  a^2+b^2=c^2
\end{equation}
がなりたつ。
```

のように書く。最初に `\documentclass[12pt,leqno]{jarticle}` のように、`leqno` (left equation number) を指定してある場合は、式番号は左側につく。

ピタゴラスの定理から
$$a^2+b^2=c^2 \tag{1}$$
がなりたつ。

5.2　かっこ

丸い括弧 (,) とカギ括弧 [,] は普通に入力できる。{, } は前に \ をつける。

```
\[
  \{[(a+b)+c]+d\}
\]
```

とすると

$$\{[(a+b)+c]+d\}$$

となる。かっこの大きさを調節するには、\left と \right で挟む場合が多い。

```
\[
  \left[
   \left(x-x_0\right)^2+\left(y-y_0\right)^2
  \right]^{1/2}
\]
```

$$\left[(x-x_0)^2+(y-y_0)^2\right]^{1/2}$$

やや脱線気味だが、最近は \left と \right の間に \middle というのを使えるようになった。

```
\[
 A=\left\{\frac{1}{n}\middle| n\in\N\right\}.
\]
```

$$A=\left\{\frac{1}{n}\middle| n\in\mathbf{N}\right\}.$$

(LaTeX2HTML では棒の背が高くならないのだけど、eplatex ではちゃんと背が高くなる。ここは実はズルをしている。)

こうすると | の前後に適当な空白が入らず、バランスが悪い。棒を高くする必要がなければ \mid を使えば良いのだが、\mid は \middle で使えない。

| を関係演算子扱いしつつ、\middle で高さを伸ばすには、次のようにすると良い (http://tex.stackexchange.com/questions/5502/how-to-get-a-mid-binary-relation-that-grows)。

```
\newcommand{\relmiddle}[1]{\mathrel{}\middle#1\mathrel{}}

\[
  A=\left\{\frac{1}{n}\relmiddle| n\in\N\right\}.
\]
```

$$A=\left\{\frac{1}{n}\relmiddle| n\in\mathbf{N}\right\}.$$

5.3 空白 (スペース)

数式モード中は、たくさんの空白用コマンドがある[12]。

```
\[
 a\,a\;a\ a\quad a\qquad a
\]
```

$$a\,a\;a\ a\quad a\qquad a$$

空白を詰めることも必要になる。\! で詰まる (マイナスの空白)。

```
\[
  \int\int f(x,y)dxdy=\int\!\!\int f(x,y)dxdy
\]
```

とすると

$$\int\int f(x,y)dxdy=\int\!\!\int f(x,y)dxdy$$

となる (左辺と右辺の積分記号の間隔を比べよう)。

(もっとも最近の TEX には、重積分・三重積分用に、\dint, \tint というコマンドが用意されているので、出番は少なくなった ??)

5.4 色々な記号

5.4.1 ギリシャ文字

\ の後にローマ字 (ラテン文字) で読みを書くことでギリシャ文字が書ける。

```
\[
  \alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi
  % omicron は o と字の形が同じなのでない
  \pi\rho\sigma\tau\upsilon\phi\chi\psi\omega
\]
```

とすると

$$\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega$$

となる。

なお、

[12]quad (=quadrat) 印刷用語で空白の込め物 (広辞苑によると、「組版の際に、印刷する必要のない余白部を埋めるために組み込むもの。」だそうである。字と字の間に入れるのが「スペース」、大きな余白にいれるのが「クワタ」であるとか。) の一種。個人的には、焼き鳥の「ハツ」 (heart) を思い出してしまう…

```
\[
  \varepsilon\vartheta\varpi\varrho\varsigma\varphi
\]
```

とすると、

$$\varepsilon\vartheta\varpi\varrho\varsigma\varphi$$

大文字のギリシャ文字は、先頭のローマ字を大文字にすればよい。例えば

```
\[
  \Gamma  \Delta  \Theta  \Lambda  \Xi  \Pi  \Sigma  \Upsilon  \Phi
  \Psi  \Omega
\]
```

とすると

$$\Gamma\Delta\Theta\Lambda\Xi\Pi\Sigma\Upsilon\Phi\Psi\Omega$$

となる (これ以外は、ローマ字の大文字と同じ。例えば α の大文字は A で良い。)。

数式で使われる文字は、字体をイタリックにする場合が多いが、ギリシャ文字の大文字をイタリックにするには、\mathit{} を用いる。

```
\[
  \mathit{\Gamma  \Delta  \Theta  \Lambda  \Xi  \Pi  \Sigma  \Upsilon  \Phi
  \Psi  \Omega}
\]
```

$$\mathit{\Gamma\Delta\Theta\Lambda\Xi\Pi\Sigma\Upsilon\Phi\Psi\Omega}$$

5.4.2 集合と論理

```
\[
  a\in A\subset B,\quad
  C\supset D,\quad
  a\not\in A,\quad
  C\not\supset D,\quad
  A\cup B, A\cap B, A\setminus B=\emptyset,\quad
  \bigcup_{i=1}^\infty A_i=\bigcap_{i=1}^\infty B_i
\]
```

$$a\in A\subset B,\quad C\supset D,\quad a\notin A,\quad C\not\supset D,\quad A\cup B, A\cap B, A\setminus B=\emptyset,\quad \bigcup_{i=1}^\infty A_i=\bigcap_{i=1}^\infty B_i$$

$\in$ (\in) の逆向きが $\ni$ (\ni) であるのは苦し紛れっぽいけど。(\supset も苦し紛れと思ったのだけれど、subset の反対語は superset なので、正しい言葉遣いなのだった。)

包含関係で等号をつけるつけないは、普通の大小関係の不等号 $<$ と同じ感じ。

```
\[
 A\subseteq B,\quad
 A\subseteqq B,\quad
 A\subsetneq B,\quad
 A\subsetneqq B.
\]
```

$$A\subseteq B,\quad A\subseteqq B,\quad A\subsetneq B,\quad A\subsetneqq B.$$

論理の記号: and $\wedge$ は \wedge, or $\vee$ は \vee, not $\neg$ は \neg とする。

```
\[
 \neg(P\wedge Q)\equiv \neg P\vee \neg Q.
\]
```

$$\neg(P\wedge Q)\equiv \neg P\vee \neg Q.$$

矢印のところで説明済みだが、$\Leftrightarrow$ は \Leftrightarrow, $\Rightarrow$ は \Rightarrow

5.5 上つき添字、下つき添字、それと積分＆シグマ

a^2 は a^2 とする。a_n は a_n とする。2^{2^n} は 2^{2^n} とする。

積分やシグマなどもこの応用で、

```
\[
  \lim_{R\to\infty}\int_a^R f(x)dx=\sum_{n=1}^\infty a_n
\]
```

とすると

$$\lim_{R\to\infty}\int_a^R f(x)dx=\sum_{n=1}^\infty a_n$$

5.6 分数

```
\[
  \frac{a+b}{c}=\frac12
\]
```

は

$$\frac{a+b}{c}=\frac{1}{2}$$

となる。

分数や積分、和の記号など、インライン数式では小さく組版されるが、ディスプレイ数式と同じように大きく組版するには、\displaystle コマンドを用いる。

```
$\frac{a+b}{c}=\frac12$ は小さいので、
$\displaystyle\frac{a+b}{c}=\frac12$ とすると大きくなる。
```

は

$\frac{a+b}{c}=\frac{1}{2}$ は小さいので、$\displaystyle\frac{a+b}{c}=\frac{1}{2}$ とすると大きくなる。

実は \dfrac という命令もある (amsmath パッケージが必要)。

なお、\displaystyle は長くて入力が面倒なので、後述するマクロなどを利用する人が多いようである。\begin{document} の前に

```
\newcommand{\dsp}{\displaystyle}
```

と定義しておくと、以下 \dsp で、\displaystyle としたのと同じになる。

TEX の分数の横棒は “短め” である。長くしたい場合は、分母か分子 (横幅の多い方) に適当なスペースを入れると良い。

```
\[
  \frac{1}{2}+\frac{1}{3}=\frac{1}{\;2\;}+\frac{1}{\;3\;}.
\]
```

$$\frac{1}{2}+\frac{1}{3}=\frac{1}{\;2\;}+\frac{1}{\;3\;}.$$

分母・分子と分数の横棒がくっつきすぎと感じることがある。分子を \raise 長さ ボックス で持ち上げ、分母を \lower 長さ ボックス で下げて微調整する（？）。

```
\[
 \frac{\kakko{ア}}{\kakko{イ}}
 =
 \frac{\raise0.8ex\hbox{$\;\kakko{ア}\;$}}
      {\lower1ex\hbox{$\kakko{イ}$}}
\]
```

$$\frac{\boxed{\text{ア}}}{\boxed{\text{イ}}}=\frac{\boxed{\text{ア}}}{\boxed{\text{イ}}}$$

5.7 sin などの「作用素」

$\boxed{\sin x}$ の ‘s’, ‘i’, ‘n’ はイタリックでない[13]、いわゆる立体 (ローマン体) で、sin と x の間に適度のスペースがあることに注意。こういうものには、専用のコマンドが用意されている場合が多い。

```
\[
 \sin x=\log y=\max A
\]
```

$$\sin x = \log x = \max A$$

単に `log x` のように書くと $logx$ となってしまう (これでは l, o, g, x の積にしか見えない)。なぜだか考えてみることを勧める。

マクロというものを使って、自分でこの種のコマンドを作ることも出来る。\begin{document} の前に (「プリアンブルに」という)

```
\newcommand{\grad}{\mathop{\mathrm{grad}}\nolimits}
```

と書いておくと (呪文のようですが、“grad” のところだけ変えれば良い、と覚えましょう)、\grad というコマンドが定義できる。

5.8 矢印

\to	\mapsto	\leftarrow	\rightarrow	\leftrightarrow	\longleftarrow	\Leftarrow
$\to$	$\mapsto$	$\leftarrow$	$\rightarrow$	$\leftrightarrow$	$\longleftarrow$	$\Leftarrow$

```
\[
  \to \quad \mapsto \quad
  \leftarrow          \quad \Leftarrow          \quad
  \longleftarrow      \quad \Longleftarrow      \quad
  \leftrightarrow     \quad \Leftrightarrow     \quad
  \longleftrightarrow \quad \Longleftrightarrow
\]
```

とすると

$$\to \quad \mapsto \quad \leftarrow \quad \Leftarrow \quad \longleftarrow \quad \Longleftarrow \quad \leftrightarrow \quad \Leftrightarrow \quad \longleftrightarrow \quad \Longleftrightarrow$$

もちろん left の反対の right もある。

上下、斜めの矢印については、

[13]普通、数式中のローマ字は、x のようにイタリック (斜めに傾いているのが特徴) で表すことに注意。

```
\[
  \uparrow \quad \downarrow \quad \Uparrow \quad \Downarrow \quad \Updownarrow \quad \Downarrow \quad
  \updownarrow \quad \Updownarrow \quad
  \nearrow \quad \nwarrow \quad \searrow \quad \swarrow
\]
```

とすると

$$\uparrow \quad \downarrow \quad \Uparrow \quad \Downarrow \quad \Updownarrow \quad \Downarrow \quad \nearrow \quad \nwarrow \quad \searrow \quad \swarrow$$

\to → と \mapsto ↦ はそのまま覚え、それ以外は命名ルールを理解して覚えることを勧める。

(nearrow, nwarrow, searrow, swarrow は、northeast, northwest, southeast, southwest だと思っているのだけど、本当かなあ？)

5.9 点

```
\[
 \cdot \quad \cdots \quad \ldots \quad \ddots \quad \vdots
\]
```

は順に、真ん中に一つの点、真ん中に３つの点、下に３つの点、斜めに３つの点、垂直方向に３つの点となる (c は center, l は low, d は diagonal (対角線の), v は vertical (垂直の))。

$$\cdot \quad \cdots \quad \ldots \quad \ddots \quad \vdots$$

5.10 不等式

等号のつかないものはそのまま $<, >$ を使うとよい。$\leq$ は \le とし、$\geq$ は \ge とする[14]。

```
\[
 a<b\le c\ge d
\]
```

$$a < b \leq c \geq d$$

なお \ll, \gg で $\ll, \gg$ となる。また、等号 $=$ の否定 $\neq$ は \ne と入力する。

平行線を含む $\leqq, \geqq$ は、それぞれ \leqq, \geqq と入力する (これは次の項で説明する AMS 拡張であるので、プリアンブルに \usepackage{amssymb} とする必要がある)。

[14]多分、“less than or equal to” から le, “greater than or equal to” から ge となったのであろう。

5.11 その他の記号

|	\pm	\mp	\times	\div	\sim	\simeq	\fallingdotseq	\leqq	\geqq
‖	±	∓	×	÷	∼	≃	≒	≦	≧

\nabla	\triangle	\partial	\forall	\exists	\infty	\propto	\angle
∇	△	∂	∀	∃	∞	∝	∠

\langle	\rangle
⟨	⟩

\fallingdotseq ≒ のような AMS (アメリカ数学会) 由来のフォントには、プリアンブルに \usepackage{amssymb} と書くことが必要です。例えば次のようにします。

```
...
\usepackage{amssymb}% AMS で用意したシンボルのフォント
...
\begin{document}
...
\[
  \| \quad \pm\quad \mp \quad \times \quad \div \quad
 \sim \quad \simeq \quad \fallingdotseq \quad \leqq \quad \geqq\quad
 \nabla \quad \triangle \quad
 \partial \quad
 \forall \quad \exists \quad
 \infty \quad \propto\quad
 \angle\quad \langle\quad \rangle
\]
...
```

$$\| \quad \pm \quad \mp \quad \times \quad \div \quad \sim \quad \simeq \quad \fallingdotseq \quad \leqq \quad \geqq \quad \nabla \quad \triangle \quad \partial \quad \forall \quad \exists \quad \infty \quad \propto \quad \angle \quad \langle \quad \rangle$$

ちなみに \fallingdotseq ≒ や \partial ∂ は長いので、筆者はマクロ (7.1 参照) を使って短い別名を定義してある [15]。

5.12 行列、ベクトル、場合分けの {

行列や (縦) ベクトルでは、式 (成分) を「きれいに並べる」必要がある。このためには、array 環境や matrix 環境を用いる (縦ベクトルは、列の個数が 1 である行列とみなす)。また括弧 (と) (あるいは [,], {, }) は \left と \right を使って拡大する。

$$\left(\begin{array}{cc} a & b \\ c & d \end{array} \right) \left(\begin{array}{c} x \\ y \end{array} \right)$$

は

[15] 解析屋にとって、偏微分記号 ∂ は良く使うので…

array 環境を用いて行列を書く

```
\[
 \left(
  \begin{array}{cc}
     a & b \\
     c & d
  \end{array}
 \right)
 \left(
  \begin{array}{c}
     x \\
     y
  \end{array}
 \right)
\]
```

または AMS 拡張に含まれる pmatrix 環境を用いても良い。

pmatrix 環境を用いて行列を書く

```
\documentclass[12pt,leqno]{jarticle}
...
\usepackage{amsmath}% プリアンブルに書く
...
\begin{document}
...
\[
  \begin{pmatrix}
     a & b \\
     c & d
  \end{pmatrix}
  \begin{pmatrix}
     x \\
     y
  \end{pmatrix}
\]
```

pmatrix 環境の方が使い方は簡単だが[16]、array 環境は左寄せ (l)、中央揃え (c)、右寄せ (r) など細かい制御ができる。

なお

$$|x| = \begin{cases} x & (x \geq 0 \text{ のとき}) \\ -x & (x < 0 \text{ のとき}) \end{cases}$$

も似た感じで出力できる。

[16]なお、括弧の形の違う行列を作る bmatrix, Bmatrix 環境、括弧なしの matrix 環境等もある。

```
\[
 |x|=
 \left\{
 \begin{array}{rl}% 1 でなく l（エル L の小文字）left の頭文字なので
   x & \text{($x\ge 0$ のとき)}\\
  -x & \text{($x<0$ のとき)}
 \end{array}
 \right. % 右側は括弧なし（ドット . が重要）
\]
```

他に cases 環境というのもあるが、右寄せ、中央寄せなど細かい指定は出来ない。

```
\[
 |x|=
 \begin{cases}
   x & \text{($x\ge 0$ のとき)}\\
   -x & \text{($x< 0$ のとき)}
 \end{cases}
\]
```

5.13 数式中の言葉

数式中に日本語や英語で説明の言葉を書きたいことがある。そういう場合は、\mbox{} や、\text{} を使う (後者は文字の大きさを回りの式に合わせて調節してくれる)。

```
\usepackage{amsmath}% \text{} に必要

...

\[
  f(x)=\log x\quad\mbox{($x$ は正の実数)},\quad
 \zeta(s)=\prod_{\text{$p$ は素数}}\frac{1}{1-\dfrac{1}{p^s}}
\]
```

$$f(x)=\log x \quad (x \text{ は正の実数}), \quad \zeta(s)=\prod_{p \text{ は素数}} \frac{1}{1-\dfrac{1}{p^s}}$$

5.14 数式の縦揃え

複数行に渡る数式を書く場合、等号など適当な位置で揃えたくなることがある。色々なやり方があるが、とりあえず align, align* 環境を覚えておきましょう。

等号の位置を揃える

```
\begin{align*}
 x&=1,\\
 f(x)&=10.
\end{align*}
```

$$x = 1,$$
$$f(x) = 10.$$

行の先頭を揃える

```
\begin{align*}
 &x=1,\\
 &f(x)=10.
\end{align*}
```

$$x = 1,$$
$$f(x) = 10.$$

アスタリスク (*) なしの align 環境は数式番号がつく。

等号の位置を揃える (数式番号つき)

```
\begin{align}
 x&=1,\\
 f(x)&=10.
\end{align}
```

(2) $$x = 1,$$

(3) $$f(x) = 10.$$

(ちなみに単独の式で式番号をつけるには、\[と \] の代りに \begin{equation} と \end{equation} (equation 環境) を用いる。

```
\begin{equation}
  3^2+4^2=5^2.
\end{equation}
```

(4) $$3^2 + 4^2 = 5^2.$$

5.15 下線、上線、矢印など

```
\[
  \underline{ABC}, \quad
  \overline{ABC}, \quad
  \overrightarrow{ABC}, \quad
  \overleftarrow{ABC}, \quad
  \overbrace{ABC}, \quad
  \underbrace{ABC}
\]
```

$$\underline{ABC},\quad \overline{ABC},\quad \overrightarrow{ABC},\quad \overleftarrow{ABC},\quad \overbrace{ABC},\quad \underbrace{ABC}$$

\overbrace{} で上に注釈をつけたい場合は、いわゆる上付き添字としてやれば良い。

```
\newcomamnd{\R}{\mathbb{R}}
\[
  \R^n:=\overbrace{\R\times\cdots\times\R}^{\text{$n$個}}
\]
```

$$\mathbb{R}^n := \overbrace{\mathbb{R}\times\cdots\times\mathbb{R}}^{n\text{ 個}}$$

5.16 misc

- $\overset{\text{def.}}{=}$ はどうやって出しますか? 昔は \stackrel{}{}, 今だと \overset{}{} を使えだとか。

```
\[
  f(x)\stackrel{\mathrm{def.}}{=}x^2+2x+3,\quad
  g(x)\overset{\mathrm{def.}}{=}3x^2+2x+1,\quad
  h(x)\underset{\mathrm{def.}}{=}\sin x.
\]
```

$$f(x)\overset{\text{def.}}{=}x^2+2x+3,\quad g(x)\overset{\text{def.}}{=}3x^2+2x+1,\quad h(x)\underset{\text{def.}}{=}\sin x.$$

- $\lim\limits_{\substack{y=kx\\(x,y)\to(0,0)}}$ は上 \atop 下 や \genfrac{}{}{0pt}{1}{ 上 }{ 下 }

```
\[
  \lim_{y=kx\atop (x,y)\to(0,0)}\frac{x y}{x^2+y^2}
  =
  \lim_{\genfrac{}{}{0pt}{1}{y=kx}{(x,y)\to(0,0)}}\frac{x y}{x^2+y^2}
\]
```

$$\lim_{\substack{y=kx\\(x,y)\to(0,0)}}\frac{xy}{x^2+y^2}=\lim_{\substack{y=kx\\(x,y)\to(0,0)}}\frac{xy}{x^2+y^2}$$

6　文書の構造など

6.1　chapter, section, subsection, paragraph など

ある程度以上長い文書は、章や節などのまとまりがある。LaTeX では、以下のような命令がある。

\part{ 見出し }　　第 x 部
\chapter{ 見出し }　　第 x 章
\section{ 見出し }　　第 x 節
\subsection{ 見出し }
\subsubsection{ 見出し }
\paragraph{ 見出し }　　段落
\subparagraph{ 見出し }

jarticle スタイルでは、part, chapter は使用できない (大抵のレポートは section で十分のはず)。chapter が使いたい場合、jreport や jbook スタイルを用いる。part が使いたい場合、jbook スタイルを用いる。

jbook スタイルを使うには…最初に指定する

```
\documentclass[12pt,leqno]{jbook}
```

6.2　自動目次生成

前項の命令を使って、chapter や section を作っていれば

```
\tableofcontents
```

という命令で自動的に目次を生成できる— 非常に便利であり、活用することが強くお勧め出来る。

6.3　参考文献表

レポートや論文では、参考にした文献や論文を文書の末尾に並べたリストを作るのが普通である。

LaTeX で参考文献表を作る方法はいくつかあるが (私は普段は pbibtex を使っている)、ここでは、もっとも単純な方法を紹介しよう。

次の例では、2 冊の本からなる参考文献表を作成してある。

```
\begin{thebibliography}{99}% 99 はとにかくこう書く
\bibitem{奥村美文書}
  奥村晴彦，\LaTeXe\ 美文書作成入門 改訂第 5 版，技術評論社 (2010).
\bibitem{クヌース}
  ドナルド・E. クヌース著，鷺谷 好輝訳，
  \TeX\ ブック --- コンピューターによる組版システム，
  アスキー (1992).
\end{thebibliography}
```

こんなふうに出来上がる

参考文献

[1] 奥村晴彦, LaTeX 2ε 美文書作成入門 改訂第 5 版, 技術評論社 (2010).

[2] ドナルド・E. クヌース著, 鷺谷 好輝訳, TeX ブック — コンピューターによる組版システム, アスキー (1992).

例えば、最初の本を引用するには、\cite{ } コマンドを用いて、

```
奥村 \cite{奥村美文書} は、日本語による\TeX\ の定番の解説書である。
```

のようにする。

出来上がりの例

奥村 [1] は、日本語による TeX の定番の解説書である。

jreport, jbook クラスで thebibliography 環境を使うと、「関連図書」という題目になる。これを「参考文献」等，好きなものに変えるには、

```
\renewcommand{\bibname}{参考文献}
```

のように \bibname の再定義をすれば良い。

6.4 索引

(準備中)

6.5 この節で解説した項目の使用例

例

```
\documentclass[12pt]{jarticle}
\usepackage[a4paper]{geometry}

\begin{document}

\title{\TeX\ によるレポートの書き方}
\author{2年16組 99番 \quad 桂田 祐史}
\date{2013年5月1日}
\maketitle

\tableofcontents

\section{はじめに}
最初はこんな風に「はじめに」や「序」などの見出しのイントロを用意する。

\section{\TeX\ の解説本}
現在 \LaTeX\ を使うための定番の解説書は、奥村 \cite{奥村美文書} である。

\TeX\ の開発者自身による解説としては、クヌース \cite{クヌース} がある。
基本的な設計思想を知りたい場合は必読書であるが、現在は購入が困難である。

\section{まとめ}
レポートや論文の最後は、
「まとめ」や「結論」や「将来の課題」などで締めるのが普通である。

必要最低限のことを覚えたら、後はどんどん使ってみるのが良い。
我流に陥らないように、あまり遅くならないうちに、
一度詳しい人に見てもらって添削してもらうのがお勧め。

\begin{thebibliography}{99}
\bibitem{奥村美文書}
  奥村晴彦, \LaTeXe\ 美文書作成入門 改訂第5版, 技術評論社 (2010).
\bibitem{クヌース}
  ドナルド・E. クヌース著, 鷺谷 好輝訳,
  \TeX\ ブック --- コンピューターによる組版システム,
  アスキー (1992).
\end{thebibliography}
\end{document}
```

出来上がりは http://nalab.mind.meiji.ac.jp/~mk/labo/text/sample.pdf で確認出来る。

1ページの文書なので、目次のありがたみがピンと来ないかも知れないが (1, 2, 3 節とも開始ページは 1 なので)。

6.6 書き足すべきこと

- 相互参照 \label{} と \ref{}
  式や章・節、定理の番号など、TEX が自動的につける番号については、\label{ 文字列 } でラベルをつけておいて、後で \{ 文字列 } で参照出来ます。

- 脚注 (フットノートの作り方)　\footnote{}

7 TEX のマクロ機能、パッケージ機能の紹介

7.1 マクロ

既に紹介したように、プリアンブルに

gradient 作用素の記号を定義する

```
\newcommand{\grad}{\mathop{\mathrm{grad}}\nolimits}
```

と書いておくと、\grad というコマンドが定義できる。これは TEX のマクロという機能を使っている。

マクロは、簡単な部分だけでも、便利に使うことが出来る。例えば \displaystyle コマンドや \varepsilon コマンドのように、長くて入力が面倒なコマンドに、短い別名をつけるために使うことが出来る。そのためには、プリアンブルに例えば

\displaystyle, \varepsilon を手短に \dsp, \eps で

```
\newcommand{\dsp}{\displaystyle}
\newcommand{\eps}{\varepsilon}
```

のように書けば良い。

マクロでは、いわゆる引数を用いることができる。2×2 の行列 $\left(\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \right)$ は、例えば

```
\left(
\begin{array}{cc}
 1 & 2 \\
 3 & 4
\end{array}
\right)
```

として組版できるが、

2×2 行列用のマクロ

```
\newcommand{\gyouretsu}[4]{
 \left(
 \begin{array}{cc}
  {#1} & {#2}\\
  {#3} & {#4}
 \end{array}
 \right)
}
```

とマクロ \gyouretsu を定義しておくと (行列の 4 つの成分が引数として与えられる)、

```
\gyouretsu{1}{2}{3}{4}+\gyouretsu{5}{6}{7}{8}=
\gyouretsu{6}{8}{10}{12}
```

で $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}$ が組版できる。

なお、マクロの名前には、ローマ字のみが使えます (gyouretu22 のような文字列は使えません)。

実は、通常使っている LaTeX そのものが、膨大なマクロの集成に他なりません。

7.2 パッケージ

LaTeX である程度まとまったことをやりたい場合に、パッケージというものが用意されていることがある (中身は要するにマクロの集合である)。

パッケージは、プリアンブルで \usepackage{} コマンドを用いて読込む。

詳細は省略するが (自分で必要になってから調べれば良い)、以下筆者が良く利用しているものの名前をあげておく。

geometry パッケージ TeX 文書で使う**紙の大きさや、余白の長さ**などを指定するのに、geometry パッケージ[17] というものが便利である (latex geometry.ins で geometry.sty を生成する)。

```
\usepackage[a4paper]{geometry}
```

のように使う。

amsmath, amssymb パッケージ **複雑な数式や、やや珍しい記号類**の組版には、アメリカ数学会 (American Mathematical Society, AMS) が開発した amsmath, amssymb パッケージが威力を発揮する。

```
\usepackage{amsmath,amssymb}
```

graphicx パッケージ **グラフィックス**を取り込むための \includegraphics{} 命令が用意されている (使い方は後述する)。

```
\usepackage[dvipdfmx]{graphicx}

あるいは

\usepackage[dvips]{graphicx}% 昔は dvips を使っていたので
```

LaTeX Beamer パッケージ **プレンゼンテーション資料**を TeX で作るために、色々なパッケージが開発されている。LaTeX Beamer パッケージはその一つである。このあたりは流行り廃りがあるので、自分が必要になったときに、WWW で検索すると良い。§11 を見よ。

[17] http://tug.ctan.org/tex-archive/macros/latex/contrib/geometry/

ascmac パッケージ 円記号を組版する \yen や、枠で囲う screen 環境、見出しつきの枠で囲う itembox 環境などは、ascmac パッケージにある。

```
\usepackage{ascmac}
```

8 ソースプログラム等テキストファイルの LaTeX 文書への取り込み

(ここは書き換えるつもりです。)

例えばプログラミングがらみの課題のレポートを作る場合など、ソースプログラムやプログラムの実行結果を取り込みたくなる。

短いものは

verbatim 環境の利用

```
\begin{verbatim}
#include <stdio.h>
int main(void)
{
  printf("Hello, world\n")
  return 0
}
\end{verbatim}
```

のように、.tex ファイルの中の、verbatim ("verbatim" は「言葉通りに」、「逐語的に」という意味の単語) 環境の中に入れてしまえばよいが、長いものや頻繁に変更を加えるものを扱うのは面倒である。

そういうものは別途テキスト・ファイルにして、**moreverb** パッケージを組み込むと有効になる \verbatimtabinput{} コマンドや \listinginput{}{} コマンド (行番号つき) を使って取り込むとよい。

hello.c, world.c を取り込む

```
\documentclass[12pt,leqno]{jarticle}
\usepackage{moreverb}% パッケージを組み込む

\begin{document}
...
\verbatimtabinput[4]{hello.c}% hello.c は別途用意してあるとして。4 カラムタブ
...
\listinginput{1}{world.c}% world.c は別途… 行番号を 1 から振る
...
\end{document}
```

念のため、以前勧めていた verbatimfiles パッケージの使い方を書いておく。

古いです！　以前は moreverb の代わりに verbatimfiles を使っていました

verbatimfiles パッケージを組み込むと有効になる \verbatimfile{} コマンドや \verbatimlisting{} コマンド (行番号つき) を使うとよいでしょう。

hello.c を取り込む

```
\documentclass[12pt,leqno]{jarticle}
\usepackage{verbatimfiles}% パッケージを組み込む（複数形の s がついている）

\begin{document}
...
\verbatimfile{hello.c}% hello.c は別途用意してあるとして
...
\end{document}
```

verbatimfiles.sty というファイルが必要ですが、例えば http://nalab.mind.meiji.ac.jp/~mk/labo/tex/style/verbatimfiles.sty から入手して、.tex ファイルと同じフォルダ (ドキュメントの下にある syori2 という人が多いはず) に置いて下さい。具体的には、マウスカーソルをリンクに合わせて、マウスを右クリックして、「名前をつけてリンク先を保存」あるいは「対象をファイルに保存 (A)」を選択します (Internet Explorer では、普通に表示した後に、[ファイル] メニューから [名前をつけて保存] で保存するには、[テキストファイル (*.txt)] 形式を選択して保存し、保存がすんでから verbatimfiles.sty という名前に変更する必要があります。面倒で間違えやすいので、ここでは右クリックして保存する方法を推奨します。)

9　画像の LaTeX 文書への取り込み

(書き換え中)

9.1　概要

LaTeX は、多くの人達の努力により、色々なグラフィックス・データを取り込めるようになっている。

具体的に何が出来るかは使用する印刷・表示用のドライバーに依存し、対応状況は結構頻繁に変化している。かなり良くなっていて、もう少しで誰でもトラブル・フリーで出来るようになる、その一歩手前だろうか。運が悪いと「はまる」かもしれないが、そこでめげないように。

(1) ドライバーを指定するオプションは最初に指定しておくのが良さそうである (ドライバーは他のパッケージとも関係するため、一番上でやっておくのが、混乱が生じにくい)。ドライバーの種類として、dvips, dviout, dvipdfm, dvipdfmx など色々ある。ずっと以前は Windows では dviout, UNIX では dvips というのが多かったが、最近の日本語環境では dvipdfmx を使うのが良いようだ。

```
\documentclass[12pt,...,dvipdfmx]{jarticle}
```

のように \documentclass{} のオプションで指定する。

(2) グラフィックス取り込み用のパッケージとして、graphics, graphicx があるが、とりあえず graphicx で良い。

```
\usepackage{graphicx}
```

(3) 画像ファイルを取り込みたいところで、

```
\includegraphics[オプション]{ファイル名}
```

とする。

- 画像ファイルは、.tex ファイルと同じディレクトリィか、その下に作ったサブディレクトリィに置くと良い。
- 細かい注意： ファイル名は日本語を避ける方が無難。特に Mac OS X のファイル名の文字コードは、UTF8 の Normalization form D というもので、今のところ色々問題を引き起こす種になっている。自分で理解して克服するつもりがない限り、日本語を避けよう。
- ドライバーとして dvipdfmx を使う場合、取り込めるファイルのフォーマットは、JPEG (.jpg), PNG (.png), PDF (.pdf), EPS (.eps) など、色々ある。その他のフォーマットであっても、これらのどれかに変換することは難しくないので、実際上困ることはないと言って良い。
- ドライバーとして dvips を使う場合は、直接取り込めるファイルのフォーマットは、EPS (.eps) だけであるが、JPEG は jpeg2ps というコマンドで EPS にラップしてから取り込むことが出来る。
- includegraphics のオプションには、height= (高さ指定), scale= (倍率指定), angle= (回転角度指定), clip (はみ出した部分を切り取る), bb= (BoundigBox 情報の指定) などがある。回転する場合の原点の指定 origin= (指定できるのは c, tl, tr, bl, br)
- 画像の大きさ (BoundingBox 情報) は、EPS の場合は内部に BoundingBox コメントとして含まれている場合が多い。$[x_1, x_2] \times [y_1, y_2]$ の場合 %%BoundingBox: x_1 y_1 x_2 y_2 とする。

  BoundingBox コメントの例

  ```
  %%BoundingBox: 36 295 595 841
  ```

  JPEG, PNG, PDF の場合は、TEX の設定がきちんとされていれば自動的に取得される。それ以外に、includegraphics のオプションで

  bb=左座標 右座標 下座標 上座標　　単位はポイント (?)

  のように直接指定することも可能である。

  こんなふうに直接 BoundingBox 情報を与えられる

  ```
  \includegraphics[width=10cm,bb=0 360 0 375]{photo0620.png}
  ```

- includegraphics 命令で取り込んだ図は、figure 環境で配置するのが望ましい。

画像ファイルの BoundingBox 情報の自動取得の設定　ここでは、少し前までの相場を説明する。(もうすぐ以下に書いてあることを意識する必要はなくなる見込みだが…)

以下、myimage.png を取り込む場合で説明する。.png のところは .pdf, .jpg などでも同様である。

myimage.png の BoundingBox 情報を得るため、TEX は外部のプログラムの力を借りて、BoundingBox 情報を書き込んだ myimage.xbb というファイルを生成し、TEX はそれを読み込んで必要な空白を作り、実際の画像の埋め込みはドライバー・プログラムに任せる、という処理の流れになっている。

実際は extractbb という外部プログラム (実は実体は dvipdfmx) を用いていた。手動で myimage.xbb を作るには、ターミナルから

```
extractbb mygraph.png
```

のように実行する。

これを自動化するために、設定ファイル texmf.cnf の中の shell_escape_commands= に extractbb を含めておく。

texmf.cnf の shell_escape_commands= の設定例

```
shell_escape_commands = \
bibtex,bibtex8,bibtexu,pbibtex,upbibtex,biber,\
kpsewhich,\
makeindex,mendex,texindy,\
mpost,pmpost,upmpost,\
repstopdf,epspdf,extractbb,\

```

\ は行継続を表すので、最後に少なくとも 1 つの空行が必要である。

最近の TeXLive 環境では、texmf.cnf は /usr/local/texlive/texmf-local/web2c/ に置くのが良いとされている。自分で作らない限り存在しないので、初めて作った場合は (上の枠内の 7 行だけの内容の texmf.cnf とすれば良い)

```
sudo mktexlsr
```

を実行して、/usr/local/texlive/texmf-local/web2c/texmf.cnf が加わったことを教える必要がある。

注意すべき点

- 画像ファイルを途中で myimage.png から (例えば) myimage.pdf に変えた場合、myimage.xbb は作り直しになる。その場合は手動で作り直すか、古い myimage.xbb を削除する必要がある。
- TeXLive 2014 の LATEX では、.xbb を生成しないようになっている (どういう仕組みで BoundingBox を得るのか、現時点で理解していない)。その場合でも myimage.xbb があればそれを読むので、古いものを掃除しておく必要がある。

9.2 PostScript データの取り込み

画像ファイルには色々なフォーマットがあるが、PostScript は古くからレーザープリンター用の言語として使われているもので、問題が生じにくかった。

LaTeX に取り込む場合は、**カプセル化 PostScript 形式** (Encapsulated PostScript, 長いので EPS 形式と呼ぶことにする, 通常は “.eps” という拡張子をつける) に変換しておくのが良い。

最近は Mathematica の出力する EPS ファイルが巨大なものとなったり、そもそも表示印刷するためのソフトが OS 標準で用意されていないこともあって[18]、必ずしもイチオシのフォーマットとは言えなくなったと思う。

kamehoshi2.eps を取り込む

```
\documentclass[12pt,leqno,dvipdfmx]{jarticle}
\usepackage{graphicx}% graphicx パッケージが必要

\begin{figure}[htbp]
 \centering
 \includegraphics[width=5cm]{eps/kamehoshi2.eps}
 \caption{星を蒔いてみる}
\end{figure}
```

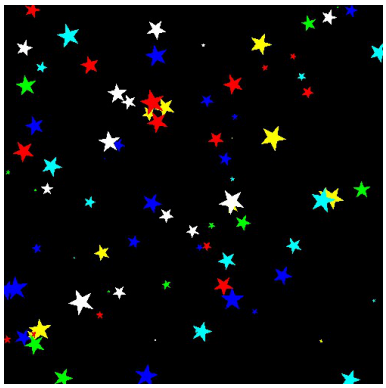


図 6: 星を蒔いてみる

- Mathematica (`http://nalab.mind.meiji.ac.jp/~mk/syori2/mathematica/node60.html`)
- gnuplot (`http://nalab.mind.meiji.ac.jp/~mk/labo/howto/intro-gnuplot/node12.html`)
- GLSC (`http://nalab.mind.meiji.ac.jp/~mk/labo/howto/intro-glsc/node26.html`)

等々では、グラフィックスを PostScript データとして出力するのは簡単である。

Mathematica の場合は,

```
g=Plot[Sin[x],{x,0,2Pi}]
Export["mygraph.eps",g]
```

のようにする。あるいは (そうして作った PostScript データが巨大になってしまう場合は)

[18] Mac で MacPorts を使っている場合は、`sudo port install gv; sudo port install ghostscript-fonts-hiragino` とすれば、Ghostscript と、それを使って表示する gv がインストールできる。

```
Export["mygraph.jpg", g, ImageResolution->1200]
```

として JPEG で出力してから (解像度を 1200 dpi にするのは好みの問題)、

```
jpeg2ps mygraph.jpg > mygraph.eps

あるいは

convert mygraph.jpg mygraph.eps
```

として PostScript に変換する。

gnuplot の場合は、

```
gnuplot> set term postscript eps color
gnuplot> set output "mygraph.eps"
```

のようにしてから描画コマンドを実行する。なお、最近は

```
gnuplot> set term push
gnuplot> set term postscript eps color
gnuplot> set output "mygraph.eps"
gnuplot> (描画コマンドを実行)
gnuplot> set term pop
```

とするのが相場かもしれない (以前は、元に戻すために、`set term x11` とか `set term win` くらい覚えておけば良かったが、最近は結構複雑なので、push, pop が用意されたらしい)。

GLSC の場合は、描画デバイスの指定時に

```
  g_init("mygraph", ...);
  g_device(G_BOTH);
  ...
```

のようにファイル (名前は `g_init()` で指定した "mygraph" になる) に出力するもの (ここでは `G_BOTH`) を選び、

```
g_out -i mygraph
```

で変換する。`mygraph.i00` というファイルが出来るが、

```
  \includegraphics[angle=90,width=10cm]{mygraph.i00}
```

のように `angle=90` で回転して取り組むか (`width=` と `angle=` の順番には注意すること)、

```
g_out -iv mygraph
```

のようにして出力時に回転する (`-v` でポートレート・モードにする、そうである)。`-v` を指定した場合、しばしば負の座標を持つ BoundingBox が出来て色々障害の原因となるが (ファイル先頭部分にあるので、`head mygraph.i00` とかしてチェックして下さい)、`ps2eps` などを用いて座標の平行移動を行なうと良い。

```
ps2eps -t=100,200 mygraph.i00

(100,200 はイイカゲンです)
```

とすると mygraph.i00.eps というファイルが生成される。元々 g_out の作る PostScript ファイルの BoundingBox 情報はイマイチなので、ps2eps はつねに実行することにした方が良いかもしれない (ある程度まともな BoundingBox に直してくれる)。

9.3 JPEG イメージの取り込み

現在のデジタルカメラの主流の画像フォーマットである JPEG データの取り込みを説明する。

写真以外でも使われる場合がある。例えば十進 BASIC のグラフィックスの場合、「名前をつけて保存 (A)」から JPEG 形式で (ファイル名拡張子は ".JPG") 保存する。

9.3.1 JPEG イメージを直接取り込む

dvipdfmx のようなドライバーを使っている場合は、直接 \includegraphics{} で取り込める。

9.3.2 JPEG イメージ PostScript に変換しての取り込み

dvips のような古いドライバーを使っている場合は、JPEG のままでの取り込みは出来ない。

しかし JPEG ファイルは、**jpeg2ps**[19] や convert (ImageMagick に含まれている) コマンドで、EPS 形式に変換してから取り込むことが可能である。

Windows 環境に Cygwin がインストールされている場合、その中に jpeg2ps が入っていることもある。コマンドプロンプトや、Cygwin のシェルで

```
Z:¥.windows2000¥syori2>jpeg2ps kamehosi.JPG > kamehosi.eps
```

とすると、kamehosi.JPG を EPS 形式に変換したファイル kamehosi.eps が出来る。

Mac ならば、MacPorts でインストールすることが出来る。

```
sudo port install jpeg2ps

使い方は上と同様である。

jpeg2ps kamehosi.jpg > kamehosi.eps
```

なお、Windows 7 の GUI で使える wjpeg2ps[20] というプログラムもある。使い方は簡単で、JPEG ファイルを wjpeg2ps のアイコンにドラッグして、convert ボタンを押すだけで EPS 形式のファイルが出来る。

仕組みについて、もう少し詳しい説明が読みたければ、「イメージデータの TEX への取り込み — jpeg2ps のすすめ」[21] を見ると良い。実際にはデータのラッピングをしているだけなので、画質の低下は生じない。

[19] http://www.pdflib.com/
[20] http://www.vector.co.jp/soft/dl/win95/art/se248407.html
[21] http://nalab.mind.meiji.ac.jp/~mk/labo/howto/jpeg2ps.html

9.4 JPEG 以外のイメージファイルの取り扱い

JPEG 以外のイメージ・ファイルのフォーマットには、Windows BMP, GIF, TIFF, PNG など色々ある。

ドライバーとして dvipdfmx を使っている場合、png や png などは直接 includegraphics 出来るわけだが、tiff などは変換する必要がある。

また dvips を使っている場合は、実質 EPS と JPEG しか読み込めないので、他のほとんどのフォーマットは変換する必要がある。

ある時期までの私のお奨めは (今は「とっとと環境を新しくして、dvipdfmx 使えるようにしましょう」がお勧め)、

最初が何であれ JPEG に変換してから、
jpeg2ps で PostScript に直して取り込む、

というやり方であった (最初が BMP だったりすると、これでかなりファイルのサイズを小さくすることができる)。二度続けて変換するのは品質を落としそうだが、実は最近の PostScript は内部に JPEG データを含むことができるようになっていて、jpeg2ps はそれをやっているだけなので、実際にデータの内容を変更するのは、最初に JPEG に変換している過程だけである。

それでは、JPEG 以外のイメージ・データをどうやって、JPEG に変換するかであるが、素の Windows であればペイントを使うのが最も簡単であろうが (名前をつけて保存のところで出力の形式が選択できる)、IrfanView などの使うのが良いと思われる (もっとも私はずっと長いこと使っていない…)。

UNIX 環境 (含む Cygwin, Mac) であれば、ImageMagick に含まれている convert が簡単である。使い方は、例えば JPEG にするのであれば

```
convert nantoka.bmp nantoka.jpg
convert nantoka.gif nantoka.jpg
convert nantoka.png nantoka.jpg
```

という感じで、出力ファイル名の拡張子を `.jpg` にするだけである。

9.5 dviout でカラー表示・印刷をするには

カラーで表示・印刷するには、dviout で Option → Setup Parameters → Graphic で、GIF の取り扱いの設定で `BMP(full-color)` を選択する。dviout 起動時に、`-GIF=5` というオプション引数を指定しても良い。これをデフォールトの設定にする人も多いが、情報処理教室のプリンターはモノクロなので、さぼってある。

9.6 余談: ウィンドウの画像を取り込む

Windows 7 のウィンドウの画像をファイルに保存したければ、マウスカーソルを取り込みたいウィンドウに置いて、キーボードから Alt+Print Screen (Print Screen は、場合によっては Fn キーと一緒に押す必要があり、その場合は Alt+Fn+PrintScreen となる) を入力し、**ペイント**[22] のようなソフトにペーストしてから、適当に編集した後で、保存すると良いでしょう (もちろん JPEG 形式に出来る)。

[22] スタート → すべてのプログラム (P) → アクセサリ → ペイント として起動できる。

Mac の場合は、標準で付属しているプレビューのファイルメニューの「スクリーンショットを撮る」を用いると良い (とても使いやすい)。

追記: PDF を PS にする　もちろん convert で convert nantoka.pdf nantoka.eps とすることも出来るが、ghostscript 由来の pdf2ps が案外使いやすい。

```
pdf2ps nantoka.pdf
ps2eps nantoka.ps
```

これで nantoka.eps が出来る。結果はコンパクトで画質も良いような印象がある。PDF と PostScript は相性が良い？

9.7 misc

9.7.1 ドライバーについて

graphicx のためにドライバーの指定が必要だが、それは他のパッケージにも影響する。例えば color.sty を使うならば、そちらにも同じドライバーを指定する。

方法 1

```
\documentclass[12pt,dvipdfmx]{jarticle}

\usepackage{graphicx}
\usepackage{color}
```

方法 2

```
\documentclass[12pt]{jarticle}

\usepackage[dvipdfmx]{graphicx}
\usepackage[dvipdfmx]{color}
```

この color.sty の件は良く知られているが、他にもドライバーと関係するパッケージがある。うまく動かない場合は調べる必要がある (個人的に gouji.sty というのを使っていて、その中で \RequirePackage[dvips]{graphicx} となっていて、ひっかかる原因になった)。

ずっと長い間、方法 2 を用いていたのだが、その方法では、geometry や TikZ などが問題を引き起こすようである。現時点では、方法 1 を推奨する。

9.7.2 .xbb ファイル

mygraph.{pdf,png,jpg} を取り込むには、BoundingBox 情報を記録した mygraph.xbb というファイルを用意する必要がある。

こうやって .xbb ファイルを作る

TeXLive に入っている extractbb を用いて

```
extractbb mygraph.pdf
```

あるいは

```
xbb mygraph.pdf
```

extractbb は TeXLive に含まれているようである (実体は dvipdfmx の別名)。$TEXMF/web2c/texmf.cnf に

```
% 次は
%   t（何でも実行可能）
% か
%   p（shell_escape_commands で指定したもののみ実行可能）
shell_escape = p

shell_escape_commands = \
bibtex,bibtex8,bibtexu,pbibtex,upbibtex,biber,\
kpsewhich,\
makeindex,mendex,texindy,\
mpost,pmpost,\
repstopdf,epspdf,extractbb,\
```

のように extractbb を入れておくと、.xbb ファイルを自動生成してくれるようである — と言うのは昔の話？最近の TEX は .xbb ファイルを作らずにサイズの方法を取得している ??

extractbb も xbb もこの後で出て来る ebb も、実体は dvipdfmx のリンクであるらしい。

xbb がない場合、例えばこんな感じで準備できる

```
$ which dvipdfmx
/usr/local/texlive/2014/bin/x86_64-darwin/dvipdfmx

(→ 場所が分った。そこに cd してリンクをする。)

$ pushd /usr/local/texlive/2014/bin/x86_64-darwin/
$ sudo ln -s dvipdfmx xbb
$ popd
```

dvipdfmx.def というファイルが古いと、.xbb ファイルの自動生成が出来ないことがあった。その場合CTAN (ftp://ftp.kddilabs.jp/CTAN/macros/latex/contrib/dvipdfmx-def/dvipdfmx.def) から最新版を取得すると良い。

(メモ: 以前は \usepackage[dvipdfmx]{graphicx} でなくて、\usepackage[dvipdfm]{graphicx} だった。その場合は ebb コマンドで .bb ファイルを作成して使う。この ebb も dvipdfmx のリンクで良い。LATEX Beamer (11) が dvipdfm しか使えなかったことがあったが、今では逆に dvipdfmx オプションしか使えないようになった。dvipdfm オプションの利用に関する情報はまだ落せない。)

9.8 figure 環境

(工事準備中)

もちろん figure 環境の説明を書かないとダメ。

図は文字と比べて大きいのが、普通で、組版で位置を決めるのが難しい (論理的な順番を尊重しすぎると、大きな余白が出来たり、おかしな組版になってしまう)。同じようなものに表がある。TEX は、図については figure 環境で、表については table 環境で扱うのが良い、とされている。

ひな形としてはこんな感じ

```
\begin{figure}[htbp]
  \centering
  \includegraphics[なんとか]{かんとか}
  \caption{図の説明（いわゆるキャプション）}
  \label{fig:引用するための文字列}
\end{figure}
```

(ようやく頭に入ったと思ったら、TeXShop のテンプレートは、これとほぼ同じものをペタッと貼り付けてくれるんですね。)

ときどき、配置しない図がたまりすぎて、TEX がこけることがある。そういうときは、\clearpage で、たまっている図を吐き出す (あまりきれいな配置にならなくても、強制的に配置する)。

キャプションを複数行書きたければ ccaption パッケージを読み込んで、\legend{} を使う。

```
\usepackage{ccaption}
...
  \caption{キャプション (1 行目)}
  \legend{キャプション (2 行目)}
```

図の配置位置を TEX 任せにせずに、自分の指定した位置に出したければ、float パッケージを読み込んで [H] を使う (H は「絶対にここ (here)」という意味らしい。昔の here.sty みたいなものか？)。

```
\usepackage{float}
...
\begin{figure}[H]
   ...
\end{figure}
```

10 TikZ

TEX には、昔から picture 環境と呼ばれる図を描くための仕掛けが用意されていたが、機能がかなり限定されていて、率直に言って使いにくいものであった。そのため、別の手段を追及するようになったのだが、現在では **TikZ** が良い選択肢であるらしい。

10.1 準備

グローバルに dvipdfmx オプションを指定するのが良いようだ。

```
\documentclass[...,dvipdfmx]{jarticle}

\usepackage{graphicx}
\usepackage{tikz}
```

次善の策として

```
\usepackage[dvipdfmx]{graphicx}
\usepackage{tikz}
```

10.2 マニュアル

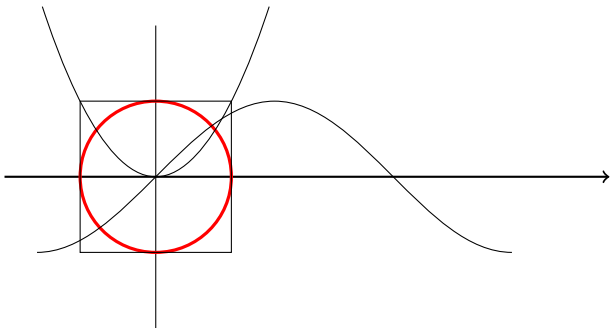
ターミナルから次のようにすればマニュアルが読める。

```
texdoc tikz
```

実体は/usr/local/texlive/2014/texmf-dist/doc/generic/pgf/pgfmanual.pdf とか。

10.3 いろは — 直線、円などを描く

```
\begin{tikzpicture}
  \draw [->,thick] (-2,0) -- (2,0);% → 付きの線分を太く
  \draw (0,-2) -- (0,2);
  \draw [very thick,red] (0,0) circle [ radius=1 ];% 中心=(0,0), 半径=1
  \draw (-1,-1) rectangle (1.0,1.0); % 左下=(-1,-1), 右上=(1,1)
  \draw (0,0) parabola (1.5,2.25);
  \draw (0,0) parabola (-1.5,2.25);
  \draw (-1.57,-1) cos (0,0) sin (1.57,1) cos (3.14, 0) sin (4.71,-1);
\end{tikzpicture}
```



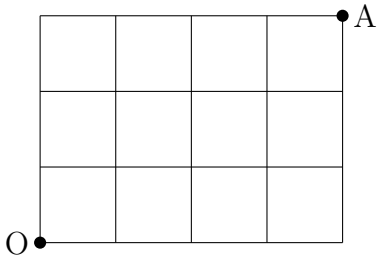
parabola は「TeX に直接作図しよう！3 」[23] で調べた。軸が垂直線の放物線の、頂点から指定した点までの範囲を描画する。

細かい工夫が色々可能である。頻繁に出て来る「重要な」点は、\coordinate コマンドで、名前をつけて参照することが出来る (同時にラベルを書くことも可能)。

\foreach で繰り返しを指定することも可能である (格子を描くのに便利だ)。

[23] http://hitgot.org/archives/drawing-in-tex-by-tikz-3/

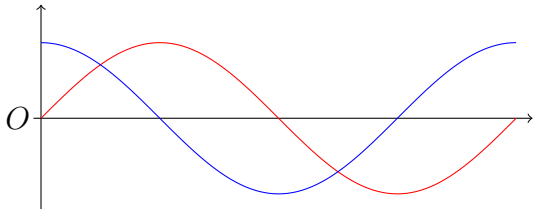
```
\begin{tikzpicture}
 \coordinate [label=left: {$\mathrm{O}$}] (O) at (0,0);
 \coordinate [label=right:{$\mathrm{A}$}] (A) at (4,3);
 \foreach \x in {0,1,2,3,4} \draw (\x,0)--(\x,3);
 \foreach \y in {0,1,2,3} \draw (0,\y)--(4,\y);
 \fill (O) circle [radius=0.08];
 \fill (A) circle [radius=0.08];
\end{tikzpicture}
```



10.4 plot

plot という命令で曲線 (折れ線？) が描ける。
座標を記録したファイルを用意しておいて plot file {ファイル名}; とすることも出来る。

```
\begin{tikzpicture}[=>stealth]
 \draw node (0,0) [left] {$O$};
 \draw [->] (-0.1,0) -- (6.5,0);
 \draw [->] (0,-1.2) -- (0,1.5);
 \draw [red] plot file {sin.tbl};
 \draw [blue] plot file {cos.tbl};
\end{tikzpicture}
```



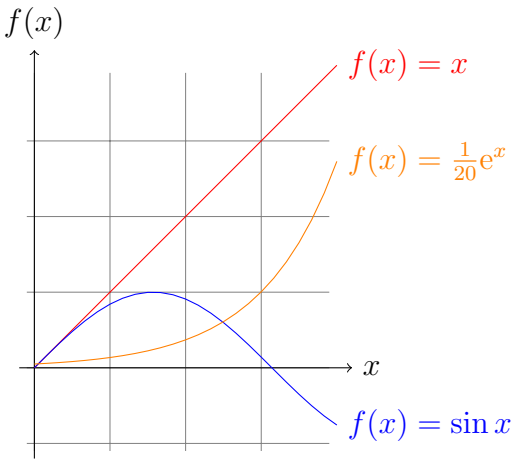
筆者はここで使っている sin.tbl, cos.tbl を、C 言語で書いたプログラムを利用して用意したが、簡単な関数の値データならば、gnuplot を利用して作成できる。

```
\begin{tikzpicture}[domain=0:4]
 \draw[very thin,color=gray] (-0.1,-1.1) grid (3.9,3.9);
 \draw[->] (-0.2,0) -- (4.2,0) node[right] {$x$};
 \draw[->] (0,-1.2) -- (0,4.2) node[above] {$f(x)$};
 \draw[color=red] plot[id=x] function{x} node[right] {$f(x) =x$};
 \draw[color=blue] plot[id=sin] function{sin(x)} node[right]
   {$f(x)=\sin x$};
 \draw[color=orange] plot[id=exp] function{0.05*exp(x)} node[right]
   {$f(x) = \frac{1}{20} \mathrm e^x$};
\end{tikzpicture}
```

これで一度組版すると、なんとか.x.gnuplot, なんとか.sin.gnuplot, なんとか.exp.gnuplot というファイルが出来る。それぞれ gnuplot で実行する。

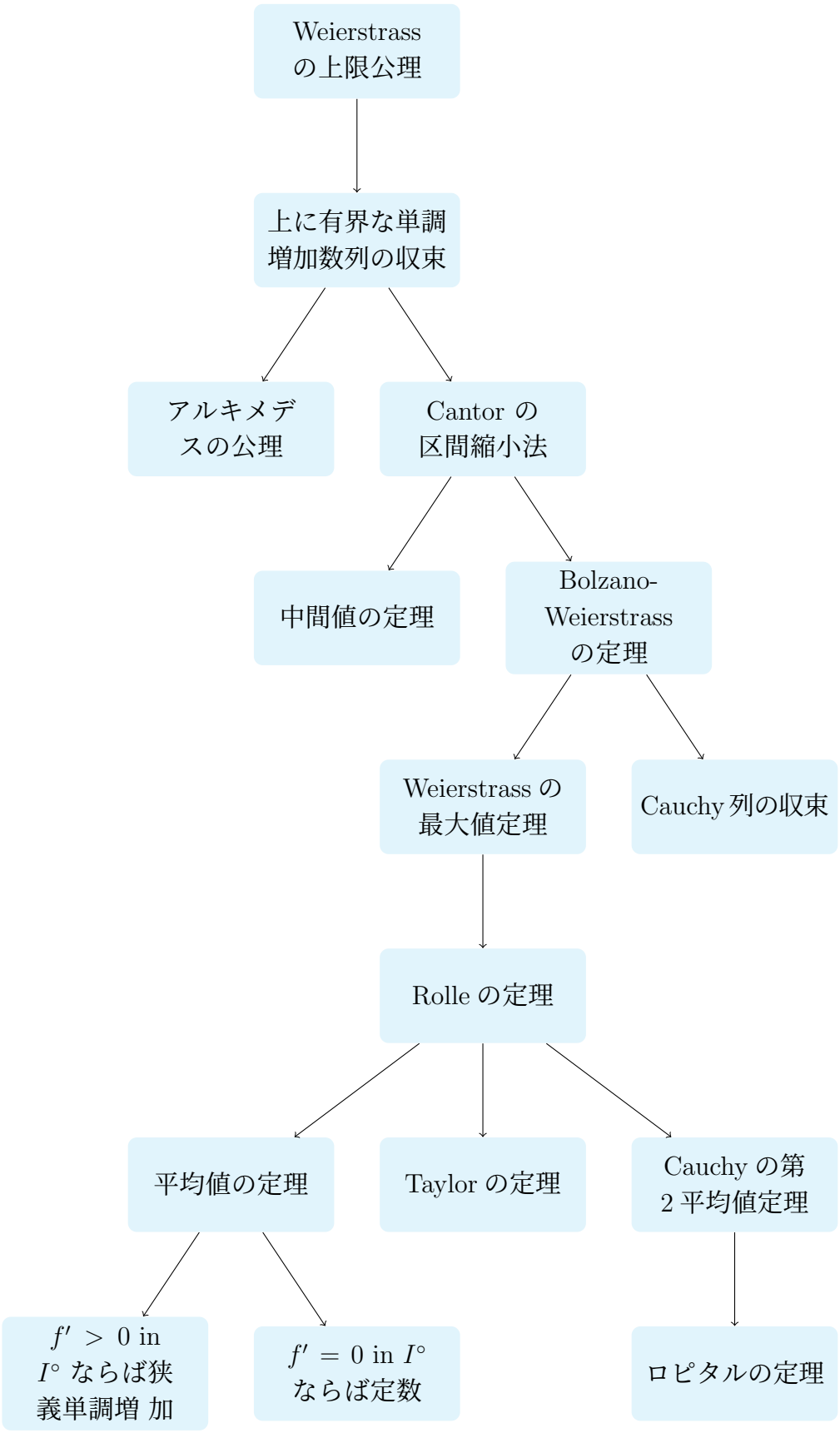
```
gnuplot なんとか.x.gnuplot
gnuplot なんとか.sin.gnuplot
gnuplot なんとか.exp.gnuplot
```

するとなんとか.x.table, なんとか.sin.table, なんとか.exp.table というファイルが出来る。もう一度組版することで作図される。



10.5 模式図

```
\begin{tikzpicture}
  \tikzset{block/.style={rectangle, fill=cyan!10, text width=3cm,
           text centered, rounded corners, minimum height=1.5cm}};
  \node[block] {Weierstrass の上限公理}
    [level distance=3cm, sibling distance=4cm,
     edge from parent/.style={->,draw}]
      child {
        node[block]{上に有界な単調増加数列の収束}
        child {
           node[block]{Cantor の区間縮小法}
             [level distance=3cm, sibling distance=4cm,
              edge from parent/.style={->,draw}]
           child{
              node[block]{中間値の定理}
           }
           child {
               node[block]{Bolzano-Weierstrass の定理}
               child {
                   node[block]{Weierstrass の最大値定理}
                   child {
                      node[block]{Rolle の定理}
                      child {
                         node[block]{平均値の定理}
                         child {
                            node [block] {$f'>0$ in $I^\circ$ ならば狭義単
調増加}
                         }
                         child {
                            node [block] {$f'=0$ in $I^\circ$ ならば定数}
                         }
                      }
                      child {
                         node[block]{Taylor の定理}
                      }
                      child {
                         node[block]{Cauchy の第 2 平均値定理}
                      }
                   }
               }
               child {
                   node[block]{Cauchy 列の収束}
               }
           }
        }
    };
\end{tikzpicture}
```



11 LaTeX Beamer でプレゼン

コンピューターの画面出力をスクリーンに映してプレゼンするのが普通になりました。Windows だと PowerPoint, Mac だと Keynote というソフトが有名ですが、TeX を使うことも出来ます。数学関係の講演では、日頃から TeX に慣れている、数式を多用する、などの理由か

ら TEX を使うのが普通です。

スクリーンは、

- 横長であるのが普通
- 同時に 1 つしか使えないのが普通
  (なるべく 1 ページに 1 話題を書き切り、1 ページに 1〜3 分の時間をかけて説明するのが良い)
- 小さい字をびっしり使うと読みづらく、大きい文字で箇条書きの文体を使うのが良い
  (この点は黒板や OHP (オーバーヘッドプロジェクター) と事情が似ている)
- 写真やカラーの図、動画が映せる、音も出せる

などの特徴があります。

TEX でそれに合った文書を簡単に作れるように、専用のスタイル・ファイルが色々開発されています。LATEX Beamer は最近人気があるスタイル・ファイルです。

11.1 準備

最近 (これを書いているのは、2014 年 2 月) は TeX Live を入れるだけで、LaTeX Beamer がちゃんと動くのが普通らしい。

11.2 必要最小限の知識

1. 最初に

   ```
   \documentclass[dvipdfmx,cjk]{beamer}
   ```

   と書く。以前は dvipdfmx でなくて、dvipdfm でないと駄目だった。今やってみて dvipdfmx で動かなかったら、LaTeX Beamer や TeX Live を更新したり、(まだ TeX Live を使っていない場合) TeX Live に乗り換えることをお勧めします。

2. 体裁の大枠は theme を指定することでなされる。

   ```
   \usetheme{Madrid}
   ```

   Madrid 以外に他にどういうテーマがあるか知りたい人、凝りたい人はネットで調べよう。

3. もちろん \begin{document} と \end{document} も必要。

4. 1 枚 1 枚をフレームと呼ぶが、それを書くのに普通は次のようにする。

   ```
   \begin{frame}{フレームの見出し}

   \end{frame}
   ```

5. dvi でなく PDF を作る。秀丸＋祝鳥ならば「PDF に変換して表示」を選択する。TeXShop を使っていれば何も意識する必要はない。

- 表題で \subtitle{} (副題) や \institute{} (所属) が使える。

```
\title{有意義な卒研のすごし方}
\subtitle{〜 楽しく真面目にやろう 〜}
\author{明治太郎}
\institute{明治大学現象数理学科}
\date{2014 年 9 月 18 日}
\frame{\titlepage}
```

- \color{ 色の名前 } や \textcolor{ 色の名前 }{ テキスト } などが使える。これらは本来 beamer のコマンドというわけではないが有益である。
- \usepackage{graphicx} を省略して、\includegraphics{} コマンドも使える。

sample.tex

```
\documentclass[dvipdfmx,cjk]{beamer}
\usetheme{Madrid}

\begin{document}

\title{有意義な卒研のすごし方}
\subtitle{〜 楽しく真面目にやろう 〜}
\author{明治 太郎}
\institute{明治大学現象数理学科}
\date{2014 年 9 月 18 日}
\frame{\titlepage}

\begin{frame}{はじめに}
  Beamer で出来ることを説明します..
\end{frame}

\end{document}
```

11.3 stepwise viewing

1 枚のスライドでスペースバーを押す度に一歩ずつ表示を進めて行く機能

- \pause が基本
- \only<数>{ テキスト } のように特定のページでだけ表示
- \uncover<数>{ テキスト } のように特定のページでだけ表示 (それ以外のページでは空白)
- \temporal<数>{ テキスト 1}{ テキスト 2}{ テキスト 3} あるページ以前、そのページ、そのページ以後
- itemize, enumerate 環境では \item<ページ指定> が使える。ページ指定としては、3 や 2,3 や 1-3,5 や 3- や -3 などの指定が出来る。

- \textcolor<2-4>{red}{ テキスト } は、2 ページから 4 ページ目でだけ赤
- 印刷時に handout オプションを指定すると、スライドが 1 枚にまとまる。

```
\documentclass[dvidfpmx,cjk,handout]{beamer}
```

11.4 リンク

その文書内のフレームにリンクを張ったボタンが作れる。

```
\hyperlink{ラベル}{\beamergotobutton{ボタンのテキスト}}
```

(ボタンをクリックするとジャンプ出来るが、戻るには Mac の Acrobat では command+←, Preview では command+[とする)

ここで言うラベルとは、次のようにしてフレームにつけることが出来る。

```
\begin{frame}[label=文字列]
```

11.5 しおりの文字化けの防止

PDF にしおりをつけるのは便利であるが、UTF8 を使って書くと、しおりが文字化けすることがある。その対処法。

beamer でしおりを付ける [24]

```
\ifnum 42146=\euc"A4A2 \AtBeginDvi{\special{pdf:tounicode EUC-UCS2}}
\else
\AtBeginDvi{\special{pdf:tounicode 90ms-RKSJ-UCS2}}
\fi
```

11.6 その他

- オプションは dvipdfm でなくて dvipdfmx に移行中?最近の MacTeX 環境などで dvipdfm を選択すると、'dvipdfm.def' not found と呼ばれることがある。
- 文字に色をつける \usepackage{color} は不要である (Beamer 自身が読み込むから)。
- グラフィックスを取り込む \usepackage{graphicx} は不要である (Beamer 自身が…)。特に

```
\documentclass[dvipdfmx,cjk]{beamer}
...
\usepackage[dvipdfmx]{graphicx}% この行は不要
```

のようにするとエラーになる。graphicx のオプション [dvipdfmx] を削除すれば通るが、そもそも \{graphicx} そのものが不要である。

[24] http://refluster.blogspot.jp/2010/10/blog-post.html

12　工事中

(定理環境、式番号の扱い等はもう少し後で。急いでいる人は、奥村 [1] のような書籍を購入したり、ネットで調べたりして下さい。)

参考文献

[1] 奥村晴彦, LaTeX 2_ε 美文書作成入門 改訂第 6 版, 技術評論社 (2013).
2010 年の第 5 版以来 3 年ぶりの新版。

[2] TeX Wiki, http://oku.edu.mie-u.ac.jp/~okumura/texwiki/

[3] ドナルド・E. クヌース著, 鷺谷 好輝訳, TeX ブック — コンピューターによる組版システム, アスキー (1992).

A　Tips

A.1　用紙のサイズ

A.1.1　LaTeX 文書の中で

まず TeX 自体に用紙のサイズを指示するには，geometry.sty で指定するのが簡単です。

```
\usepackage{a4paper}{geometry}
```

b4paper というのもありますが，これは ISO 規格だそうで，日本で普通に B4 という場合は，b4j とするのが良いのかもしれません。

日本で B4 (JIS 規格) を使う

```
\usepackage{b4j}{geometry}
```

横置きにするには landscape とします。

A4 横置き

```
\usepackage{landscape}{geometry}
```

A.1.2　後で dvipdfmx を使うことを見越して

海外では pdflatex が普及していて，.tex から直接 .pdf を作るそうですが，日本ではまだ dvipdfmx を利用して，.dvi 経由で .pdf を作るのが普通だと思います。

dvipdfmx では，-p オプションで用紙サイズの指定が出来ますが，その指定を .tex ファイルの中に埋め込むことも出来るそうです (ISO 規格にしか対応していない dvipdfmx で B4 を扱いたい時とか便利かも)。

B4 縦 (JIS)

```
\AtBeginDvi{\special{pdf: pagesize width 257mm height 364mm}}
```

B4 横 (JIS)

```
\AtBeginDvi{\special{pdf: pagesize width 364mm height 257mm}}
```

A4 横

```
\AtBeginDvi{\special{landscape}}
```

結局、B4 縦のときは geometry と合わせてこんな感じに設定

```
\usepackage[b4j,hscale=0.80,vscale=0.90]{geometry}
\AtBeginDvi{\special{pdf: pagesize width 257mm height 364mm}}
```

用紙サイズ	長い辺	短い辺
A3	420 mm	297 mm
A4	297 mm	210 mm
A5	210 mm	148 mm
B4	364 mm	257 mm
B5	257 mm	182 mm

A.1.3 色々なコマンドでの用紙サイズ指定のオプション

これは大変だ (こんなもの覚えられるわけがない！PDF の中に埋め込みたがるわけだ)。

コマンド	オプション	
xdvi	`-paper`	landscape は a4r のような名前
dvips	`-t`	landscape は `-t landscape`
dvipdfmx	`-p`	bx ($x = 0, 1, \cdots, 10$) は ISO 規格，bxj が JIS 規格, landscape は `-l`

A4 landscape の場合

```
xdvi -paper a4r myreport.dvi &
dvips -t landscape myreport.dvi | lp
dvipdfmx -l myreport.dvi
```

B4 の場合

```
xdvi -paper b4 myreport.dvi &
dvips -t b4 myreport.dvi | lp
dvipdfmx -p b4j myreport.dvi
```

B4 landscape の場合

```
xdvi -paper b4r myreport.dvi &
dvips -t b4 -t landscape myreport.dvi | lp
dvipdfmx -p b4j -l myreport.dvi
```

A.2 バージョンが書いていない PostScript ファイル

PostScript ファイルの先頭行は

```
%!PS-Adobe-1.0
```

のように、PostScript のバージョンを示す「注釈行」が入っているのが普通である。ところが

古いソフトが生成した PostScript ファイルには、このバージョン情報が空のものがあり、これが問題を引き起こすことがある (PDF を作ったとき、図が表示されないなど)。

こういうときは、嘘でもバージョン番号を書くと、うまく行くことがある

A.3 負の座標を含む BoundingBox を直す

負の座標を含んだ BoundingBox を持つ PostScript ファイルは、色々な問題を引き起こす。座標をずらすことで問題が解決できることがある。

```
ps2eps -t=100,200 kasanari.eps

(kasanari.eps.eps という名前のファイルが出力される。)
```

A.4 今いつでしょう？

現在の日付は \today で出力される (元号が嫌ならば事前に \ 西暦 としておく)。\the\year, \the\month, \the\day という LaTeX コマンドがある。pLaTeX には \the\hour, \the\minute もある。

A.5 jobname

その TeX ファイルの名前は？\jobname が使える。

A.6 MacOS プレビューメモ

TeX と直接の関係はないが、プレビューでプレゼンテーションをすることがあるので。

- ページ番号ジャンプは option+command+G
- 戻るには command+[
- 全画面表示するには control+command+F

A.7 QED

証明の終わりは amsthm パッケージを使っていれば,

```
\begin{proof}
\end{proof}
```

で勝手にやってもらえる。

自分で使うには

```
\usepackage{amssymb}% \Box に必要
\newcommand{\qed}{\hfill$\Box$}% 右隅に白抜きの四角
\newcommand{\anqed}{\hbox{\rule{6pt}{6pt}}}% 黒い正方形
```

A.8 $\mathbb{R}$ など黒板太字

$\mathbb{R}$, $\mathbb{C}$, $\mathbb{N}$, $\mathbb{Z}$ などの，太字を黒板上で二重線で表現するフォントは \mathbb{} で出力可能である。

```
\usepackage{amssymb}

..

実数体 $\mathbb{R}$
```

以前は \Bbb というコマンドを使うことになっていたが、それは obsolete (時代遅れ) であるらしい。

A.9 下付きチルダ

accents パッケージの \undertilde

A.10 ベクトルの太字

長いこと

```
\newcommand{\Vector}[1]{{\mbox{\boldmath$#1$}}}
```

というマクロを使ってきたけれど、添字が小さくならない。

```
\[
  \Vector{n}_{\Vector{y}}
\]
```

として

$$\boldsymbol{n}_{\boldsymbol{y}}$$

となるとか。

```
\usepackage{bm}
...
\[
   \Vector{n}_{\bm y}
\]
```

とすれば

$$\boldsymbol{n}_{y}$$

A.11　ベクトルの矢印

矢印は、$\vec{a}$ (\vec a) だと小さ過ぎ、$\overrightarrow{a}$ (\overrightarrow a), では大き過ぎる、あるいは $\overrightarrow{AB}$ (\overrightarrow{AB}) の矢印は開き過ぎている、など違和感を持つ場合が多い。

esvect パッケージの \vv コマンドというものがある。

```
\usepackage{esvect}
...
\begin{align*}
  &\vv{a}+\vv{b}=\vv{c},\\
  &\vv{\mathstrut a}+\vv{\mathstrut b}=\vv{\mathstrut c},\\
  &\vv{\mathstrut AB}+\vv{\mathstrut BC}=\vv{\mathstrut AC}.
\end{align*}
```

$$\begin{aligned}
&\vec{a}+\vec{b}=\vec{c},\\
&\vec{a}+\vec{b}=\vec{c},\\
&\overrightarrow{AB}+\overrightarrow{BC}=\overrightarrow{AC}.
\end{aligned}$$

A.12　rsfs フォント (ある 1 つの花文字)

Ralph Smith's formal script パッケージを使うと、$\mathscr{D}$, $\mathscr{I}$ のような花文字が利用できる。分野によって「これがなければ」というような文字があるので、重宝する。

A.13　下線

下線を引くコマンドとして、\underline{} というのがあるが、複数行に渡るような長い部分には下線を引けない。jumoline.sty の提供する \Underline{} コマンドが使えるかも知れない。

```
\usepackage{jumoline}
\setlength{\UnderlineDepth}{3pt}

\Underline{下線を引きたいところはこのようにすれば良い。}
```

参考: 大石 勝「下線に関するマクロ比較」[25]

B　日本の数学書、学校数学のルール

B.1　図形の点

図形の点をローマ字 A, B, C, ··· で表すことが多いが、そのときの字体はイタリックでなく、立体である。混同しやすいが、角 (かく) はイタリックである。

[25] http://www17.plala.or.jp/ohishi-masaru/tex/library/underline.pdf

△ABC で、∠ABC を単に角 B という。

B.2 等号の否定

$=$ の否定は TEX では、\ne とすると説明されるが ($\neq$ が出力される)、少なくても日本の学校数学の教科書では、斜め線が左上から右下に走る $\neq$ の形をしている。

```
\def\Noteq{\mathrel{%
\setbox0\hbox{=}\hbox{=}\llap{\hbox to\wd0{\hss$\backslash$\hss}}}}
```

B.3 初等幾何の記号

図形の合同は、日本では △ABC ≡ △PQR のように ≡ で表すが、英語文化圏では △ABC ≅ △PQR のように ≅ (\cong) を用いて表す。

同様に図形の相似は、日本では △ABC ∽ △PQR のように ∽ で表すが、英語文化圏では △ABC ∼ △PQR (\sim を使った) や $A \Bumpeq B$ (\mathrel{\text{\rotatebox{90}{$\equiv$}}} を使った) で表すそうである。